P-3564

ENERGY EFFICIENT AND QOS BASED ROUTING PROTOCOL FOR WIRELESS SENSOR NETWORKS

PROJECT REPORT

Submitted by

M.MOHAMED RIZWAN

Reg . No: 0920108008

In partial fulfillment for the award of the degree of

MASTER OF ENGINEERING

in

COMPUTER SCIENCE AND ENGINEERING

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE – 641 049

APRIL 2011

KUMARAGURU COLLEGE OF TECHNOLOGY

(An Autonomous Institution Affiliated to Anna University of Technology, Coimbatore)

COIMBATORE – 641 049

PROJECT WORK

Department of Computer Science and Engineering

APRIL 2011

This is to certify that the project entitled

ENERGY EFFICIENT AND QOS BASED ROUTING PROTOCOL FOR WIRELESS SENSOR NETWORKS

is the bonafide record of project work done by

M.MOHAMED RIZWAN

Register No: 0920108008

of M.E. (Computer Science and Engineering) during the year 2010-2011.

[illegible] [illegible]

Project Guide Head of the Department

Submitted for the Project Viva-Voce examination held on 25/04/2011

[illegible] 25/4/11 [illegible] 25/4/2011

Internal Examiner External Examiner

DECLARATION

I affirm that the project work titled **ENERGY EFFICIENT AND QOS BASED ROUTING PROTOCOL FOR WIRELESS SENSOR NETWORKS** being submitted in partial fulfillment for the award of M.E Computer Science and Engineering is the original work carried out by me. It has not formed the part of any other project work submitted for the award of any degree or diploma, either in this or any other University.

[illegible]
MOHAMED RIZWAN.M
Register No: 0920108008

I certify that the declaration made above by the candidate is true to the best of my knowledge.

[illegible]
Mr. K.SIVAN ARUL SELVAN, M.E.,
Assistant Professor
Department of Computer Science and Engineering,
Kumaraguru College of Technology,(An Autonomous Institution)
Coimbatore-641 049.

KONGU ENGINEERING COLLEGE

(Autonomous)

PERUNDURAI ERODE 638 052 TAMILNADU INDIA

Estd : 1984

TÜV SÜD ISO 9001

KONGU *Assuring the Best*

School of Computer Technology and Applications

Department of Computer Technology - UG

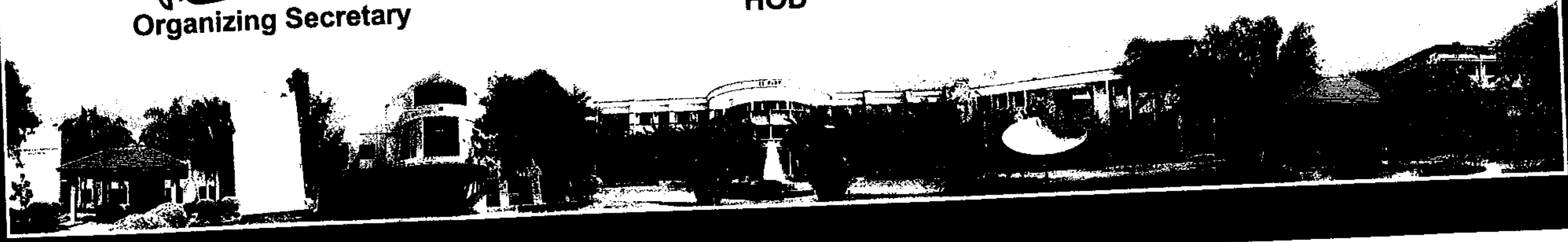
CERTIFICATE

This is to certify that Mr./~~Ms.~~ M.MOHAMED RIZWAN of KUMARAGURU COLLEGE OF TECHNOLOGY has ~~Participated~~ / Presented a paper entitled ENERGY EFFICIENT BASED ROUTING PROTOCOL FOR WIRELESS SENSOR NETWORK in the National Conference held from 10th to 11th March 2011 on Emerging Computing and Communication Technologies (NECCT 2011).

[illegible]
Organizing Secretary

[illegible]
HOD

[illegible]
Principal



ACKNOWLEDGEMENT

First and foremost, I would like to thank the Lord Almighty for enabling me to complete this project.

I express my profound gratitude to our Chairman **Padmabhusan Arutselvar Dr.N.Mahalingam, B.Sc., F.I.E.,** for given this opportunity to pursue this course.

I would like to thank **Dr.S.Ramachandran, Ph.D.,** *Principal* for providing the necessary facilities to complete my project.

I take this opportunity to thank., **Mrs.P.Devaki, M.E** , *Head of the Department,* Computer Science and Engineering, for her precious suggestions.

I thank all project committee members for their comments and advice during the reviews. Special thanks to **Mrs.V.Vanitha, M.E.,** *Associate Professor*, Department of Computer Science and Engineering, for arranging the brain storming project review sessions.

I register my hearty appreciation to the Guide **Mr. K.Sivan Arul Selvan, M.E.,** *Assistant Professor,* Department of Computer Science and Engineering. I thank for his support, encouragement and ideas. I thank for the countless hours he has spent with me, discussing everything from research to academic choices.

I would like to convey my honest thanks to all **Teaching** staff and **Non Teaching** staffs of the department for their support. I would like to thank all my classmates who gave me a proper lighter moments and study breaks apart from extending some technical support whenever I needed them most.

I dedicate this project work to my **parents** for no reasons but feeling from bottom of my heart, without their love this work wouldn't be possible.

TABLE OF CONTENTS

ABSTRACT

The increasing demands for real-time applications in Wireless Sensor Networks (WSNs) have made the Quality of Service (QoS) based communication protocols an interesting and hot research topic. Satisfying Quality of Service (QoS) requirements (e.g. bandwidth and delay constraints) for the different QoS based applications of WSNs raises significant challenges. More precisely, the networking protocols need to cope up with energy constraints, while providing precise QoS guarantee. Therefore, enabling QoS applications in sensor networks requires energy and QoS awareness in different layers of the protocol stack. In many of these applications (such as multimedia applications or real-time and mission critical applications), the network traffic is mixed of delay sensitive and delay tolerant traffic. Hence, QoS routing becomes an important issue. In this project, I propose an Energy Efficient and QoS aware multipath routing protocol (abbreviated shortly as EQSR) that maximizes the network lifetime through balancing energy consumption across multiple nodes, uses the concept of service differentiation to allow delay sensitive traffic to reach the sink node within an acceptable delay, reduces the end to end delay, and increases the throughput through introducing data redundancy. EQSR uses the residual energy, node available buffer size, and to predict the best next hop through the paths construction phase. Based on the concept of service differentiation, EQSR protocol employs a queuing model to handle both real-time and non-real-time traffic.

ஆய்வுச்சுருக்கம்

கம்பியிலா உணரி வலையமைப்பானது ஆயிரக்கணக்கான முனைகளைக் கொண்டு சுற்றுச்சூழள் சூழ்நிலைகளை கண்காணிப்பதற்காகப் பயன்படுத்தப்படுகிறது.இதன் முக்கிய குறைபாடு இதன் மின் கலத்தை நாம் மின்னூட்டு செய்யவோ அல்லது மற்றவோ முடியாது.எனவே இதன் ஆற்றலை உரிய முறையில் பயன்படுத்த வேண்டும்.

இதில் தொடர்புப்பகுதியே அதிக அற்றலை பயன்படுத்துவதாக ஆய்வுகள் தெரிவிக்கின்றன.எனவே இந்த ஆய்வில் அற்றலை திறமைமிக்க வழியில் பயன்படுத்தும் வினா வழியமைக்கும் வரைமுறை அறிமுகப்படுத்தப்படுகிறது.

இந்த இலக்கை எட்டுவதற்க்காக ஒரே சேவையை கண்காணிக்கும் முனைகள் ஒரே கூட்டமாகக் கருதப்படுகிறது.மேலும் இதற்க்கென்று ஒரு கூட்டத் தலைமுனையும் நியமிக்கப்படுகிறது.மேலும் இந்த கூட்டமைப்பானது மீதமுள்ள அற்றல் மற்றும் அடிப்படை நிலையத்திலிருந்து அதன் தொலைவு ஆகியவற்றையும் கருத்தில் கொண்டு உருவாக்கப்படுகிறது.

அடிப்படை நிலையத்தில் வினாக்கள் சமர்பிக்கப்படும்பொழுது அவைகள் கூட்டத்தலை முனை வழியாக சரியான முனைக்கு வழிப்படுத்தப்பட்டு தேவையான சேவையானது பெறப்படுகிறது.இந்த அமைப்பின் மூலம் முனைகளின் அற்றல் சேமிக்கப்படுவதுடன் அவைகளின் ஆயுட்காலமும் நீட்டிக்கப்படுகிறது.மேலும் இதில் வழியமைப்பு நகல் அட்டவணையை பேணுவதற்கான தேவை ஏற்படுவதில்லை.

LIST OF FIGURES

LIST OF ABBREVIATIONS

WSN	Wireless Sensor Networks
MCMP	Multi-Constraint Multi-Path
EQSR	Energy Efficient and QoS aware multipath Routing protocol
CBM	Condition-Based Maintenance
DTC	Distributed TCP Caching
TCP	Transmission Control Protocol
TSS	TCP Support for Sensor Networks
MEMS	Micro-Electro-Mechanical Systems
BFO	Bacterial Foraging Optimization
LR-WPAN	Low-rate Wireless Personal Area Networks
QoS	Quality of Service
SNR	Signal-to noise ratio
FEC	Forward Error Correction

LIST OF SYMBOLS

N_x	Neighbors of node x
E_{resd}	Residual Energy
$B_{buffer;y}$	Buffer size of node y,
$I_{interference;xy}$	SNR for the link between x and y
$C(_{total})$	Total Cost
$l(_{x\ y)i,l}$	Link Costs
T_r	Size of the real-time traffic
T_{nr}	Size of the non-real-time traffic

CHAPTER 1

INTRODUCTION

1.1 OVERVIEW OF WIRELESS SENSOR NETWORKS

Wireless Sensor Networks is a network of typically small, battery-powered, distributed autonomous wireless devices. A sensor node, also known as a 'mote', is a node in a wireless sensor network that is capable of performing some processing, gathering sensory information and communicating with other connected nodes in the network. In addition to one or more sensor nodes, each node in a sensor network is typically equipped with wireless communications device, a small microcontroller, and an energy source, usually a battery.

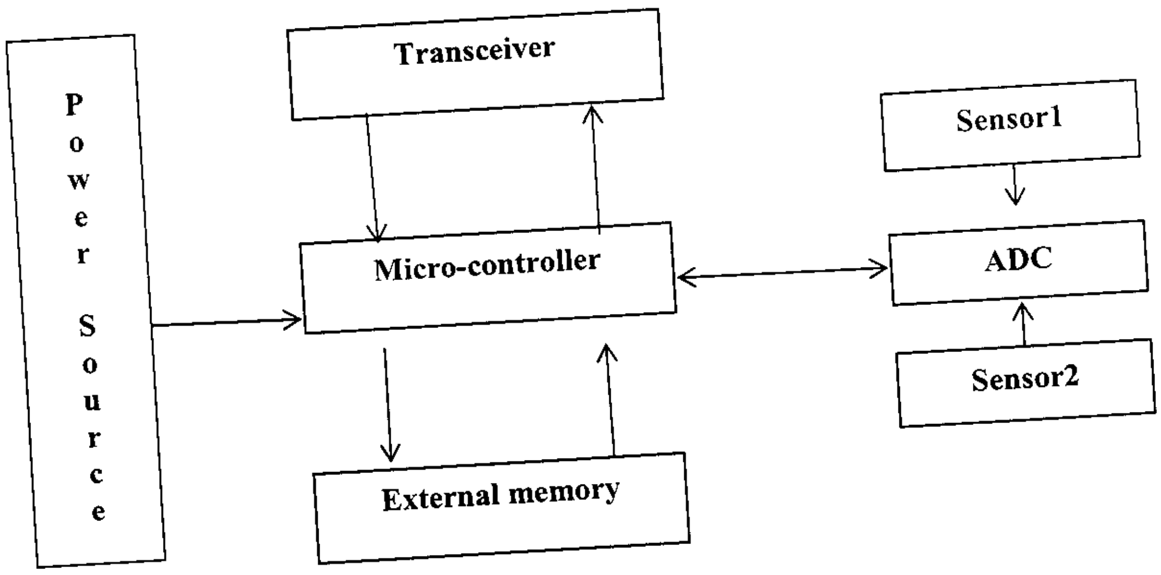


Figure 1.1 Components of Sensor Networks

- **Microcontroller**

Microcontroller performs tasks, processes data and controls the functionality of other components in the sensor node. Other alternatives that can be used as a controller are: General purpose desktop microprocessor, Digital Signal Processors, Field Programmable Gate Array and Application-specific integrated circuit. Microcontrollers are the most suitable choice for a sensor node.

- **Transceiver**

Sensor nodes make use of ISM band which gives free spectrum allocation and global availability. The various choices of wireless transmission media are Radio frequency, Optical communication (Laser) and Infrared. Laser requires less energy, but needs line-of-sight for communication and also sensitive to atmospheric conditions. Infrared like laser, needs no antenna but is limited in its broadcasting capacity. Radio Frequency (RF) based communication is the most relevant that fits to most of the WSN applications. WSN's use the communication frequencies between about 433 MHz and 2.4 GHz. The functionality of both transmitter and receiver are combined into a single device known as transceivers are used in sensor nodes. Transceivers lack unique identifier. The operational states are Transmit, Receive, Idle and Sleep.

- **External Memory**

From an energy perspective, the most relevant kinds of memory are on-chip memory of a microcontroller and Flash memory - off-chip RAM is rarely if ever used. Flash memories are used due to its cost and storage capacity. Memory requirements are very much application dependent. Two categories of memory based on the purpose of storage a) User memory used for storing application related or personal data. b) Program memory used for programming the device. This memory also contains identification data of the device if any.

- **Power Source**

Power consumption in the sensor node is for the Sensing, Communicating and Data Processing. More energy is required for data communication in sensor node. Energy expenditure is less for sensing and data processing. The energy cost of transmitting 1 Kb a distance of 100 m is approximately the same as that for the executing 3 million instructions by 100 million instructions per second/W processor. Power is stored either in Batteries or Capacitors. Batteries are the main source of power supply for sensor nodes.

- **Sensors**

Sensors are hardware devices that produce measurable response to a change in a physical condition like temperature and pressure. Sensors senses or measures physical data of the area to be monitored. The continual analog signal sensed by the sensors is digitized by an Analog-to-digital converter and sent to controllers for further processing. Characteristics and requirements of Sensor node should be small size, consume extremely low energy, operate in high volumetric densities, be autonomous and operate unattended, and be adaptive to the environment. As wireless sensor nodes are micro-electronic sensor device, can only be equipped with a limited power source of less than 0.5 Amp and 1.2 V.

- **ADC - Analog to Digital Converter**

The analog signals produced by the sensors are converted to digital signals by the analog to digital converter (ADC), and then fed into the processing unit.

1.2 APPLICATIONS OF SENSOR NETWORKS

Wireless Sensor Networks have a wide range of applications such as,

1. Military applications
    - i. Monitoring friendly forces and equipment.
    - ii. Battlefield surveillance.
    - iii. Nuclear, biological and chemical attack detection.
2. Environmental Applications
    - i. Forest fire detection.
    - ii. Bio-complexity mapping of the environment.
    - iii. Flood detection.
3. Health applications
    - i. Tele-monitoring of human physiological data.
    - ii. Tracking and monitoring doctors and patients inside a hospital.
    - iii. Drug administration in hospitals.
4. Home application
    - i. Home automation.
    - ii. Smart environment

In order to enable reliable and efficient observation and initiate right actions, physical phenomenon features should be reliably detected/estimated from the collective information provided by sensor nodes. Moreover, instead of sending the raw data to the nodes responsible for the fusion, sensor nodes use their processing abilities to locally carry out simple computations and transmit only the required and partially processed data. Hence, these properties of WSN impose unique challenges for development of communication protocols in such architecture. The intrinsic properties of individual sensor nodes, pose additional challenges to the communication protocols in terms of energy consumption.

1.2 CHALLENGES OF WIRELESS SENSOR NETWORKS

A sensor network design is influenced by many factors, which include:

- **Energy efficiency/system lifetime**

As sensor nodes are battery-operated, protocols must be energy-efficient to maximize system life time. System life time can be measured such as the time until half of the nodes die or by application-directed metrics, such as when the network stops providing the application with the desired information about the phenomena.

- **Fault Tolerance**

Some sensor nodes may fail or be blocked due to lack of power, have physical damage or environmental interference. The failure of sensor nodes should not affect the overall task of the sensor network. This is the reliability or fault tolerance issue. Fault tolerance is the ability to sustain sensor network functionalities without any interruption due to sensor node failures.

- **Scalability**

The number of sensor nodes deployed in studying a phenomenon may be in the order of hundreds or thousands. Depending on the application, the number may reach an extreme value of millions. The new schemes must be able to work with this number of nodes.

- **Production Costs**

Since the sensor networks consist of a large number of sensor nodes, the cost of a single node is very important to justify the overall cost of the networks. If the cost of the network is more expensive than deploying traditional sensors, then the sensor network is not cost-justified. As a result, the cost of each sensor node has to be kept low.

- **Environment**

Sensor nodes are densely deployed either very close or directly inside the phenomenon to be observed. Therefore, they usually work unattended in remote geographic areas. They may be working in busy intersections, in the interior of large machinery, at the bottom of an ocean, inside a twister, on the surface of an ocean. They work under high pressure in the bottom of an ocean, in harsh environments such as debris or a battlefield, under extreme heat and cold such as in the nozzle of an aircraft engine or in arctic regions, and in an extremely noisy environment such as under intentional jamming.

- **Hardware Constraints**

A sensor node is made up of four basic components: a sensing unit, a processing unit, a transceiver unit and a power unit. Sensing units are usually composed of two subunits: sensors and analog to digital converters (ADCs). The analog signals produced by the sensors based on the observed phenomenon are converted to digital signals by the ADC, and then fed into the processing unit. The processing unit, which is generally associated with a small storage unit, manages the procedures that enable the sensor node collaborate with the other nodes to carry out the assigned sensing tasks. A transceiver unit connects the node to the network. One of the most important components of a sensor node is the power unit. Power units may be supported by a power scavenging unit such as solar cells. There are also other subunits, which are application dependent. Most of the sensor network routing techniques and sensing tasks require the knowledge of location with high accuracy.

- **Sensor Network Topology**

Sheer numbers of inaccessible and unattended sensor nodes, which are prone to frequent failures, make topology maintenance a challenging task. Hundreds to several thousands of nodes are deployed throughout the sensor field.

- **Pre-Deployment Phase**

Sensor nodes can be either thrown in mass or placed one by one in the sensor field. They can be deployed by dropping from a plane, delivering in an artillery shell, rocket or missile, throwing by a catapult , placing in factory, and placing one by one either by a human or a robot. Although the sheer number of sensors and their unattended deployment usually preclude placing them according to a carefully engineered deployment plan, the schemes for initial deployment must reduce the installation cost, eliminate the need for any pre-organization and preplanning, increase the flexibility of arrangement, and promote self-organization and fault tolerance.

- **Post-Deployment Phase**

After deployment, topology changes are due to change in sensor nodes position, reach ability (due to jamming, noise, moving obstacles, etc.), available energy, malfunctioning, and task details. Sensor nodes may be statically deployed. However, device failure is a regular or common event due to energy depletion or destruction. It is also possible to have sensor networks with highly mobile nodes. Besides, sensor nodes and the network experience varying task dynamics, and they may be a target for deliberate jamming. Therefore, sensor network topologies are prone to frequent changes after deployment.

- **Re-Deployment of Additional Nodes Phase**

Additional sensor nodes can be re-deployed at any time to replace the malfunctioning nodes or due to changes in task dynamics. Addition of new nodes poses a need to re-organize the network. Coping with frequent topology changes in an ad hoc network that has myriads of nodes and very stringent power consumption constraints requires special routing protocols.

- **Transmission Media**

In a multi-hop sensor network, communicating nodes are linked by a wireless medium. These links can be formed by radio, infrared or optical media. To enable global operation of these networks, the chosen transmission medium must be available worldwide. One option for radio links is the use of *Industrial, Scientific and Medical* (ISM) bands, which offer license free communication in most countries.

- **Power Consumption**

The wireless sensor node, being a microelectronic device, can only be equipped with a limited power source. In some application scenarios, replenishment of power resources might be impossible. Sensor node lifetime, therefore, shows a strong dependence on battery lifetime. The malfunctioning of few nodes can cause significant topological changes and might require rerouting of packets and re-organization of the network. Hence, power conservation and power management take on additional importance. It is for these reasons that researchers are currently focusing on the design of power aware protocols and algorithms for sensor networks. In sensor networks, power efficiency is an important performance metric, directly influencing the network lifetime. Application specific protocols can be designed by appropriately trading off other performance metrics such as delay and throughput with power efficiency. The main task of a sensor node in a sensor field is to detect events, perform quick local data processing, and then transmit the data. Power consumption can hence be divided into three domains: *sensing*, *communication*, and *data processing*.

CHAPTER 2

LITERATURE REVIEW

2.1 Comparative Study of Wireless Sensor Networks Energy-Efficient Topologies and Power Save Protocols

The paper [2], it addresses issues associated with control of data transmission in wireless sensor networks (WSN) with stationary nodes. Since the WSN nodes are typically battery equipped, the primary design goal is to optimize the amount of energy used for transmission. The energy conservation techniques and algorithms for computing the optimal transmitting ranges in order to generate a network with desired properties while reducing sensors energy consumption are discussed and compared through simulations. It describes a new clustering based approach that utilizes the periodical coordination to reduce the overall energy usage by the network.

The paper provides the short overview of the energy conservation techniques and algorithms for calculating energy-efficient topologies for WSNs. The efficiency of four location based approaches, i.e., two schemes for topology control and two power save algorithms are discussed based on the results of simulation experiments. The energy efficient method of introducing a coordinator to a WSN is presented. It show that our algorithm outperforms the results obtained for popular clustering based power save protocol .

2.2 Energy Efficient Adaptive Multipath Routing for Wireless Sensor Networks

Routing in wireless sensor networks is a demanding task. This [3], it explain about the demand has led to a number of routing protocols which efficiently utilize the limited resources available at the sensor nodes. All these protocols typically find the minimum energy path. In this paper we

take a view that, always using the minimum energy path deprives the nodes energy quickly and the time taken to determine an alternate path increases. Multipath routing schemes distribute traffic among multiple paths instead of routing all the traffic along a single path. Two key questions that arise in multipath routing are how many paths are needed and how to select these paths. Clearly, the number and the quality of the paths selected dictate the performance of a multipath routing scheme. We propose an energy efficient adaptive multipath routing technique which utilizes multiple paths between source and the sink, adaptive because they have low routing overhead. This protocol is intended to provide a reliable transmission environment with low energy consumption, by efficiently utilizing the energy availability and the received signal strength of the nodes to identify multiple routes to the destination. Simulation results show that the energy efficient adaptive multipath routing scheme achieves much higher performance than the classical routing protocols, even in the presence of high node density and overcomes simultaneous packet forwarding over the nodes lying on different possible paths between the source and the sink, in proportion to their residual energy and received signal strength. The rationale behind traffic spreading is that for a given total energy consumption in the network, at each moment, every node should have spent the same amount of energy. The objective is to assign more loads to under-utilized paths and less load to over-committed paths so that uniform resource utilization of all available paths can be ensured. Multipath routing is cost effective for heavy load scenario, while a single path routing scheme with a lower complexity may otherwise be more desirable. It compare our proposed scheme with the directed diffusion [5] and flooding protocols Simulation results show that energy efficient adaptive multipath routing outperforms the traditional routing approaches in terms of network lifetime, load balancing and packet delivery ratio.

2.3 Energy-Efficient Wireless Sensor Network Design and Implementation for Condition-Based Maintenance

In [4]. New application architecture is designed for continuous, real-time, distributed wireless sensor networks. It develops a wireless sensor network for machinery condition-based maintenance (CBM) in small machinery spaces using commercially available products. It develops a hardware platform, networking architecture, and medium access communication protocol. It implements a single hop sensor network to facilitate real-time monitoring and extensive data processing for machine monitoring. A new radio battery consumption model is presented and the battery consumption equation is used to select the most suitable topology and design an energy efficient communication protocol for wireless sensor networks. A new streamlined matrix formulation is developed that allows the base station to compute the best periodic sleep times for all the nodes in the network. It combine scheduling and contention to design a hybrid MAC protocol, which achieves 100% collision avoidance by using our modified RTS-CTS (Request To Send-Clear To Send) contention mechanism known as TDMA (Time Division Multiple Access protocol).

2.4 Efficient Route Selection Strategies for Wireless Sensor Networks

Wireless Sensor Networks (WSNs) facilitate monitoring and controlling of physical environments from remote locations with the best possible accuracy. Sensor networks are wireless networks consisting of groups of small, inexpensive nodes, which collect and disseminate critical data. Also, sensor nodes have various energy and computational constraints due to their inexpensive nature and ad hoc method of deployment. Considerable research has been focused on overcoming these deficiencies through low-energy consumption schemes. Among other factors, the route selection strategy may have an impact on the sensors lifetime, and following on the network lifetime. In this paper, we study various route selection strategies that aim at prolonging the lifetime of WSNs. Also, a new route selection scheme is proposed, that increases further the network lifetime. The performance of these schemes is analyzed through simulation.

2.5 Energy-Efficient TCP Operation in Wireless Sensor Networks

In [6],it explains about applications of wireless sensor networks require connectivity to external networks to let monitoring and controlling entities communicate with the sensors. By using the TCP/IP protocols inside the sensor network, external connectivity can be achieved anywhere in the sensor network. In such IP-based sensor networks, TCP can be used for remote management and reprogramming of sensor nodes. However, the high bit error rates in multi-hop sensor networks lead to energy-inefficiencies that reduce the lifetime of the sensor network. This paper introduces and compares two approaches to support energy-efficient operation of TCP in sensor networks: Distributed TCP Caching (DTC) and TCP Support for Sensor networks (TSS). Both concepts allow intermediate sensor nodes to cache TCP segments and to perform local retransmissions in case of errors. This allows reducing the total number of packet transmissions in the sensor network when transferring data to or from a sensor node. DTC caches and immediately forwards TCP data segments, whereas TSS does not forward a cached segment until it knows that the previous segment has been successfully received by the next hop node. We show by simulation that both approaches significantly reduce the number of TCP segment and acknowledgement transmissions. Their performance differs slightly depending on the error rate. Both approaches have also slightly different needs in buffer requirements and TCP options to be supported.

2.6 Energy Efficient Routing Algorithms for Wireless Sensor Networks and Performance Evaluation of Quality of Service for IEEE 802.15.4 Networks

The [7],popularity of Wireless Sensor Networks (WSN) have increased tremendously in recent time due to growth in Micro-Electro-Mechanical Systems (MEMS) technology. WSN has the potentiality to connect the physical world with the virtual world by forming a network of sensor nodes. Here, sensor nodes are usually battery-operated devices, and hence energy saving of sensor nodes is a major design issue. To prolong the network's lifetime, minimization of energy consumption should be implemented at all layers of the network protocol stack starting from the physical to the application layer including cross-layer optimization.
In this project, clustering based routing protocols for WSNs have been discussed. In cluster-based routing, special nodes called cluster heads form a wireless backbone to the sink. Each cluster heads collects data from the sensors belonging to its cluster and forwards it to the sink. In heterogeneous networks, cluster heads have powerful energy devices in contrast to homogeneous networks where all nodes have uniform and limited resource energy. So, it is essential to avoid quick depletion of cluster heads. Hence, the cluster head role rotates, i.e., each node works as a cluster head for a limited period of time. Energy saving in these approaches can be obtained by cluster formation, cluster-head election, data aggregation at the cluster-head nodes to reduce data redundancy and thus save energy.

2.7 Agent-based Framework for Energy Efficiency in Wireless Sensor Networks

Wireless sensor networks are consisted of hundreds or thousands of small sensors that have limited resources. Energy-efficient techniques [8],are the main issue of wireless sensor networks. This paper proposes an energy efficient agent-based framework in wireless sensor networks. We adopt biologically inspired approaches for wireless sensor networks. Agent operates automatically with their behavior policies as a gene. Agent aggregates other agents to reduce communication and gives high priority to nodes that have enough energy to communicate. Agent behavior policies are optimized by genetic operation at the base station. Simulation results show that our proposed framework increases the lifetime of each node. Each agent selects a next-hop node with neighbor information and behavior policies. My proposed framework provides self-configuration, self-optimization properties to sensor nodes.

2.8 Energy-Aware QoS Control for Wireless Sensor Network

While a lot of research has been done on some important aspects of WSN [9],such as architecture and protocol design, supporting Quality of Service in WSN is still a largely unexplored research field. An application-specific QoS has been defined as the sensor network resolutions. In the paper [20], it design a novel energy-aware algorithm based on the previous work to solve the QoS problems. The periodical sleeping mechanism is introduced into the algorithm to save energy. When the optimal number sensors are achieved in initialization stage, not all of the remaining sleep state nodes need to wake up every second.
There exists many envisioned applications in WSN and their QoS requirements may be very different. It is unlikely that there will be a "one-size-fits-all" QoS support for most applications. The present research efforts related to the QoS in WSN fall into three categories: traditional end-to-end QoS, reliability assurance, and application-specific QoS.

CHAPTER 3

METHODOLOGY

3.1 Link Cost Function

The link cost function is used by the node to select the next hop during the path discovery phase. Let N_x be the set of neighbors of node x. Then our cost function includes an energy factor with appropriate weights (α).

$$\text{Next hop} = \max_{y \in N_x}\{\alpha E_{resd,y}\}$$

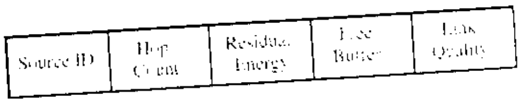
Where $E_{resd,y}$ is the current residual energy of node y, where y € N_x.

3.2 Paths Discovery Phase

Based on the idea of the directed diffusion the sink node starts the multiple paths discovery phase to create a set of neighbors that able to forward data towards the sink from the source node. The constructed multi-paths are node-disjoint paths (i.e. have no common nodes except the source and the destination). In multi-path routing, node-disjoint paths are usually preferred because they utilize the most available network resources, and hence are the most fault-tolerant. If an intermediate node in a set of node-disjoint paths fails, only the path containing that node is affected, so there is a minimum impact to the diversity of the routes [11].

The path discovery procedure is executed according to the following phases:

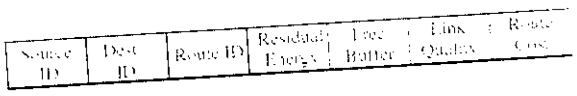
- **Initialization phase:** Each sensor node broadcast a HELLO message to its neighboring nodes in order to have enough information about which of its neighbors can provide it with the highest quality data. Each sensor node maintains and updates its neighboring table during this phase. The neighboring table contains information about the list of neighboring nodes of the sensor node. Fig. 3.2.1 illustrates the structure of the HELLO message. Hop count gives the distance in hops for the message from its originator. As the message is only broadcasted to the neighbors of the node, this field is not used in this version of the protocol.



Source ID	Hop Count	Residual Energy	Free Buffer	Link Quality

Fig 3.1 HELLO Message Structure

- **Primary Path discovery phase:** After initialization phase each sensor node has enough information to compute the cost function for its neighboring nodes. Then, the sink node locally computes its preferred next hop node using the link cost function, and sends out a RREQ message to its most preferred next hop (Fig. 3.2.2) shows the structure of the RREQ message). Similarly, through the link cost function, the preferred next hop node of the sink computes locally its most preferred next hop in the direction of the source node, and sends out a RREQ message to its next hop, the operation continues until source node.



Source ID	Dest ID	Route ID	Residual Energy	Free Buffer	Link Quality	Route Cost

Fig 3.2 RREQ Message Structure.

- **Alternative Paths discovery phase:** For the second alternate path, the sink sends alternate path RREQ message to its next most preferred neighbor. To avoid having paths with shared node, we limit each node to accept only one RREQ message. For those nodes that receive more than one RREQ message, only accept the first RREQ message and reject the remaining messages (see Fig. 3.3, for an example of path construction. In this example Node 9 computes its next preferred neighbor finds it Node 7. Node 9 generates RREQ message and forwards to node 7, but node 7 has been included in the primary path, then node 7 simply responds to node 9 with an INUSE message indicating that node 7 is already selected in a routing path. Immediately node 9, searches its neighboring table and computes the next preferred neighbor, which will be node 5, and sends out RREQ message to it. Node 5 accepts the message and continues the procedure in the direction of the source node).

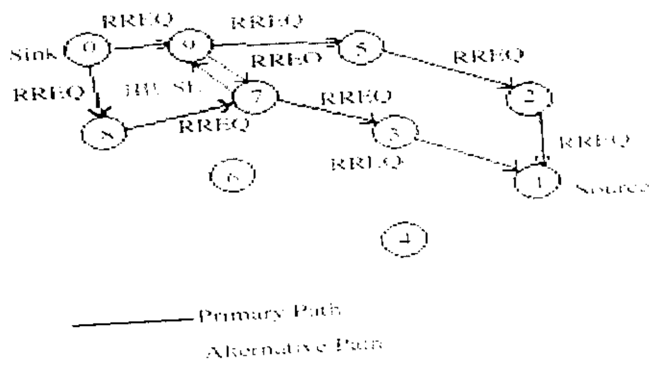


Fig 3.3 Example of Path Discovery.

3.3 Path Refreshment

In order to save energy, we reduce the overhead traffic through reducing control messages. Therefore, instead of periodically flooding a KEEPALIVE message to keep multiple paths alive and update cost function metrics, we append the metrics on the data message by attaching the residual energy, and remaining buffer size to the data message.

3.4 Paths Selection

After the completion of paths discovery phase and the paths have been constructed, we need to select a set of paths from the N available paths to transfer the traffic from the source to the destination with a desired bound of data delivery given by α. To find the number of required paths, we assume that each path is associated with some rate P $_I$ = (1,2,3..........N) that corresponds to the probability of successfully delivering a message to the destination. Following the work done in [9], the number of required paths is calculated as:

$$k = x_\alpha \cdot \sqrt{\sum_{i=1}^{N} p_i(1 - p_i)} + \sum_{i=1}^{N} p_i$$

where x $_\alpha$ is the corresponding bound from the standard normal distribution for different levels of α. Now out of the k paths, the protocol picks out a number of l paths to be used to transfer the real-time traffic, and m paths for non-real-time traffic

3.5 Traffic Allocation and Data Transmission

Our protocol employs the queuing model presented in [2] to handle both real-time and non-real-time traffic. Two different queues are used, one instant priority queue for real-time traffic, and the other queue follows the first in first out basis for non real-time traffic. Fig. 3.5.1 shows the functional diagram of the EQSR protocol

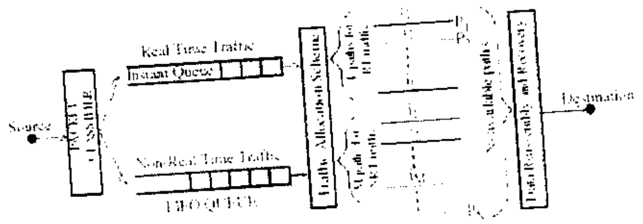


Fig. 3.5.1 Functional diagram of the EQSR protocol.

The source node knows the degree of the importance of each data packet it is sending which can be translated into predefined priority levels. The application layer sets the required priority level for each data packet by appending an extra bit of information to act as a stamp to distinguish between real-time and non-real-time packets. Based on the packet type, the classifier directs packets into the appropriate queue. The traffic allocation scheme first splits up the packets into a number of equal sized sub-packets (or segments), and then schedules sub-packets simultaneously for transmission across the available multiple paths. Before scheduling the sub-packets, the traffic allocation scheme adds error correction codes to improve the reliability of transmission and to increase the resiliency to paths failures and ensure that an essential portion of the packet is

received by the destination without incurring any delay and more energy consumption through data retransmission .At the sink node, the parts are collected, reassembled, and the original message is recovered.

CHAPTER 4

RESULTS AND DISCUSSIONS

The experiments are performed with following configuration parameters.

Parameter	Value
Network size	1500×1500(m)
Base station location	(750,750)m
Number of sensor nodes	100
Initial energy	100 J
Data packet size	1024 bytes
Transmission range	250 m

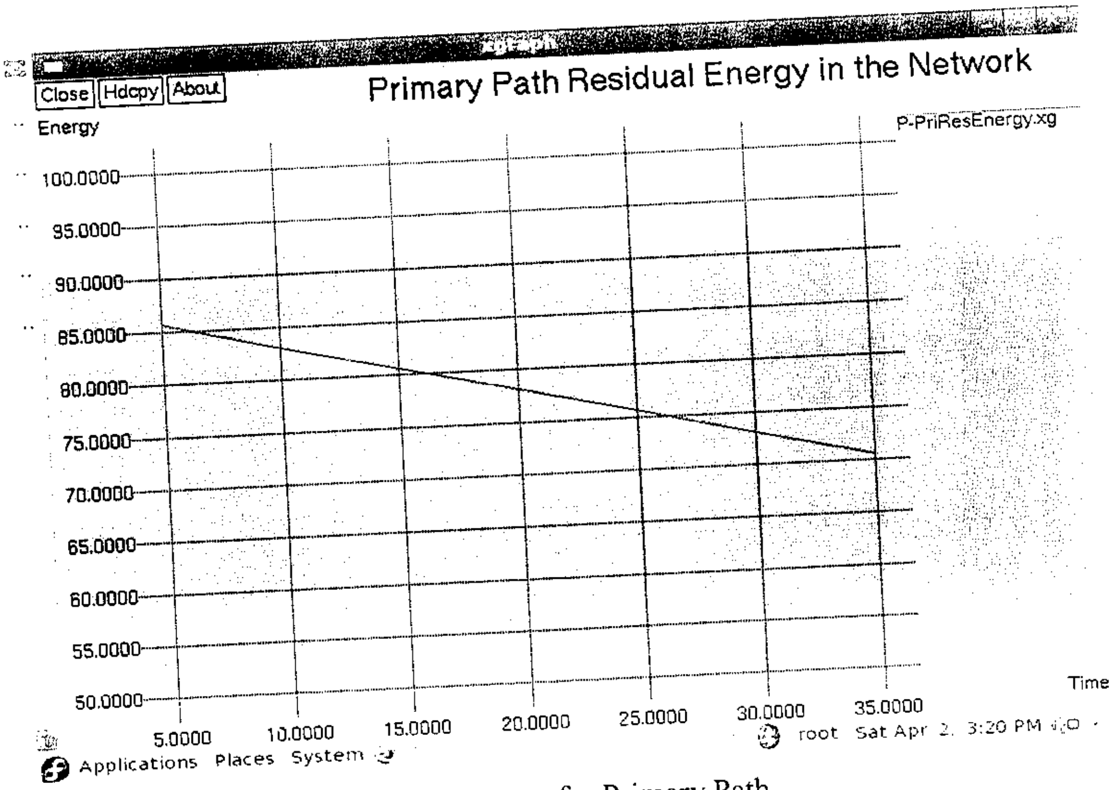


Fig 4.1 Residual Energy for Primary Path

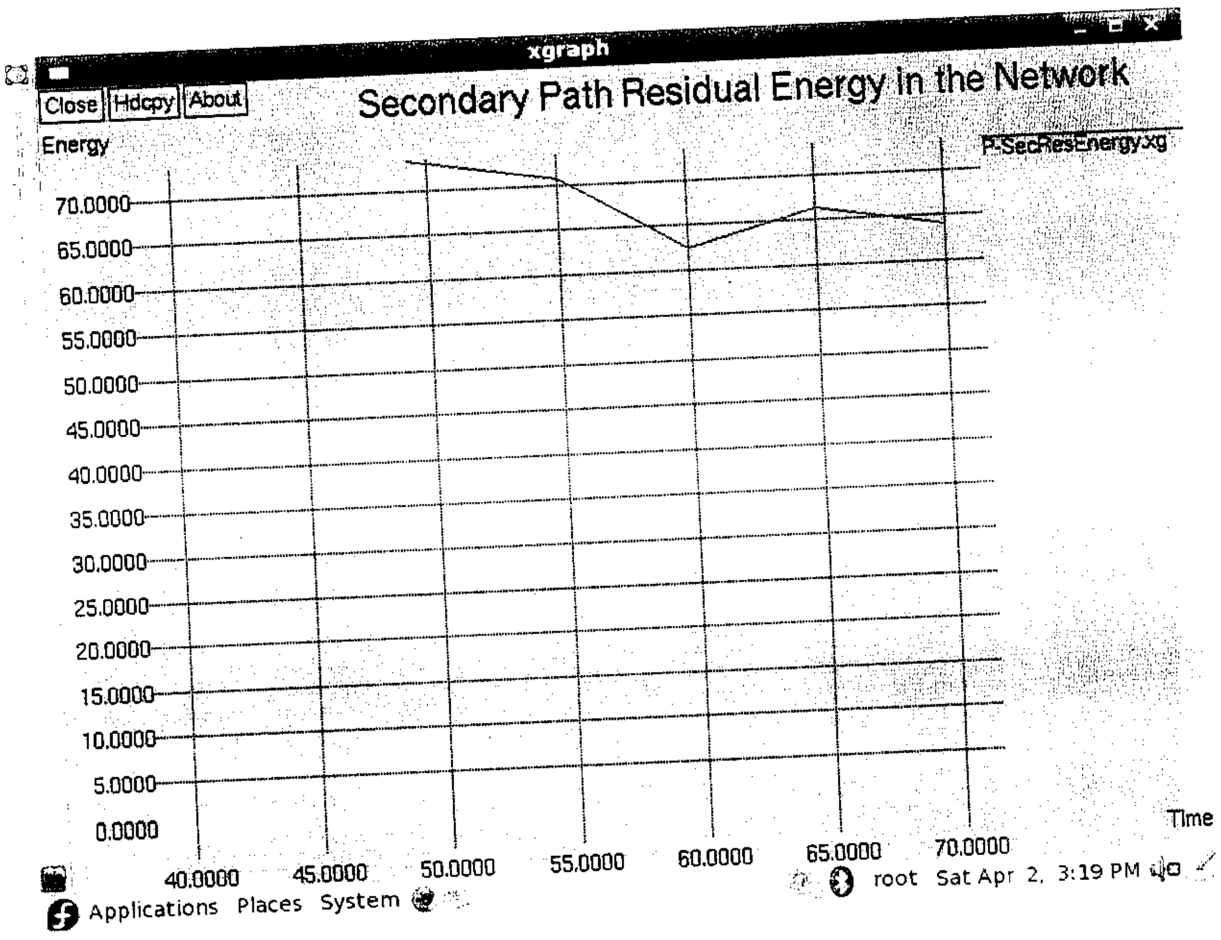


Fig 4.2 Residual Energy for Secondary Path

The performance shows of the energy graph for primary and secondary path. This shows the results for the energy consumption under node failures. Obviously our EQSR protocol outperforms the MCMP protocol in this case. EQSR protocol achieves more energy savings than MCMP protocol. This is because EQSR protocol easily recovers from path failures and be able to reconstruct the original messages through the use of the FEC algorithm. While the MCMP protocol needs to initiate a data retransmission to recover lost data, which leads to a significant increase in the energy consumption.

CHAPTER 5

CONCLUSION AND FUTURE OUTLOOK

In this project, I have presented EQSR protocol; an energy efficient and quality of service aware multi-path routing protocol designed specifically for wireless sensor networks to provide service differentiation by giving real-time traffic absolute preferential treatment over the non-real-time traffic. This protocol uses the multi-path paradigm. This feature is very important in sensor networks since flooding consumes energy and consequently reduces the network lifetime. This EQSR protocol uses the residual energy, node available buffer size to predict the next hop through the paths construction phase. EQSR protocol handles both real-time and non real-time traffic efficiently, by employing a queuing model that provides service differentiation. Through computer simulation, the protocol was evaluated and studied the performance under different network conditions. Simulation results have shown that our protocol achieves lower average delay with minimum energy requirement.

Future outlook

As a future work, more factors can be considered in the EQSR by incorporating SNR, simultaneous data transfer and study the impact of the network size, path length, and buffer size on the performance metrics.

6.APPENDIX

6.1. SOURCE CODE

Coding for Energy-Efficiency

```
              /* IMPLEMENTATION OF ENERGY-EFFICIENT */
                  /* ------------------------------------ */

# Environmental Settings
 set val(chan)        Channel/WirelessChannel  ;# channel type
 set val(prop)        Propagation/TwoRayGround ;# radio-propagation model
 set val(ant)         Antenna/OmniAntenna      ;# Antenna type
 set val(ll)          LL                      ;# Link layer type
 set val(ifq)         Queue/DropTail/PriQueue  ;# Interface queue type
 set val(ifqlen)       256                    ;# max packet in ifq
 set val(netif)        Phy/WirelessPhy         ;# network interface type
 set val(mac)           Mac/802_11             ;# MAC type
 set val(rp)               AODV                          ;# Routing Protocol
 set val(nn)          50                      ;# number of mobilenodes
 set val(x)           1500                          ;
 set val(y)           1500                          ;
 set opt(energymodel)  EnergyModel                  ;# Energy model
 set opt(radiomodel)  RadioModel                   ;# Radio model
 set opt(initialenergy)  100                    ;# Initial energy in Joules
 set val(sc)                  setdest-100.tr
 set r                        250
 set s_thres                  250
 set qlen             256
 set lc                       0
 set beta                     2
# Default parameters

  Agent/TCP/RFC793edu set rto_ 250          ;#250 m Transmission range
  Agent/TCP set packetSize_ 1024            ;# Packet size (data + overhead)
  Phy/WirelessPhy set Pt_ 0.015              ;# Transmission Power
  Phy/WirelessPhy set RXThresh_ 2.025e-12       ;#500m radius.Receving Threshold
  Phy/WirelessPhy set CSThresh_ [expr 0.9*[Phy/WirelessPhy set RXThresh_]] ;# Carrier Sence Threshold
  Phy/WirelessPhy set CPThresh_ 10.0         ;# Carrier Power
  Phy/WirelessPhy set bandwidth_ 2e6         ;# Bandwidth
  Phy/WirelessPhy set freq_ 914e+6           ;# Frequency



  # Simulator Object Creation

  set ns_ [new Simulator]
```

```
# Trace File to record all the Events


set f [open Energy-Qos.tr w]
$ns_ trace-all $f
$ns_ use-newtrace

# NAM Window creation
set namtrace [open Energy-Qos.nam w]
$ns_ namtrace-all-wireless $namtrace $val(x) $val(y)

# Topology Creation
set topo [new Topography]
$topo load_flatgrid 1500 1500

# General Operational Director
create-god $val(nn)

# Node Configuration
$ns_ node-config  -adhocRouting $val(rp) \
                  -llType $val(ll) \
            -macType $val(mac) \
            -ifqType $val(ifq) \
            -ifqLen $val(ifqlen) \
            -antType $val(ant) \
            -propType $val(prop) \
            -phyType $val(netif) \
            -channelType $val(chan) \
            -topoInstance $topo \
                  -agentTrace ON \
            -routerTrace ON \
            -macTrace ON \
            -movementTrace ON \
            -idlePower 0.012 \
                  -rxPower 1.5 \
                  -txPower 2.0 \
                  -sleepPower 0.00015 \
            -initialEnergy $opt(initialenergy) \
             -energyModel $opt(energymodel)



# Node Creation

set god_ [create-god $val(nn)]

for {set i 0} {$i < $val(nn) } {incr i} {



set node_($i) [$ns_ node]
$node_($i) random-motion 0
$god_ new_node $node_($i)
}

for {set i 0} {$i < $val(nn)} {incr i} {
        $ns_ initial_node_pos $node_($i) 40
        $node_($i) set X_ 750.0
        $node_($i) set Y_ 0.0
        $node_($i) set Z_ 0.0
        $node_($i) color black
}

source $val(sc)

# subclass Agent/MessagePassing to make it do flooding

Class Agent/MessagePassing/Flooding -superclass Agent/MessagePassing

Agent/MessagePassing/Flooding instproc recv {source sport size data} {
    $self instvar messages_seen node_
    global ns BROADCAST_ADDR

    # extract message ID from message
    set message_id [lindex [split $data ":"] 0]
    #puts "\nNode [$node_ node-addr] got message $message_id\n"

    if {[lsearch $messages_seen $message_id] == -1} {
            lappend messages_seen $message_id
        $ns trace-annotate "[$node_ node-addr] received HELLO {$data} from $source"
        $ns trace-annotate "[$node_ node-addr] sending HELLO message $message_id"
            $self sendto $size $data $BROADCAST_ADDR $sport
    } else {

    }
}

Agent/MessagePassing/Flooding instproc send_message {size message_id data port} {
    $self instvar messages_seen node_
    global ns MESSAGE_PORT BROADCAST_ADDR

    lappend messages_seen $message_id
    $ns trace-annotate "[$node_ node-addr] sending HELLO message $message_id"
    $self sendto $size "$message_id:$data" $BROADCAST_ADDR $port
}




set t [$ns_ now]

for {set i 0} {$i<50} {incr i} {
set sink$i [new Agent/LossMonitor]
}
```

```
$ns_ attach-agent $node_(0) $sink0


$ns_ attach-agent $node_(1) $sink1
$ns_ attach-agent $node_(2) $sink2
$ns_ attach-agent $node_(3) $sink3
$ns_ attach-agent $node_(4) $sink4
$ns_ attach-agent $node_(5) $sink5
$ns_ attach-agent $node_(6) $sink6
$ns_ attach-agent $node_(7) $sink7
$ns_ attach-agent $node_(8) $sink8
$ns_ attach-agent $node_(9) $sink9
$ns_ attach-agent $node_(10) $sink10
$ns_ attach-agent $node_(11) $sink11
$ns_ attach-agent $node_(12) $sink12
$ns_ attach-agent $node_(13) $sink13
$ns_ attach-agent $node_(14) $sink14
$ns_ attach-agent $node_(15) $sink15
$ns_ attach-agent $node_(16) $sink16
$ns_ attach-agent $node_(17) $sink17
$ns_ attach-agent $node_(18) $sink18
$ns_ attach-agent $node_(19) $sink19
$ns_ attach-agent $node_(20) $sink20
$ns_ attach-agent $node_(21) $sink21
$ns_ attach-agent $node_(22) $sink22
$ns_ attach-agent $node_(23) $sink23
$ns_ attach-agent $node_(24) $sink24
$ns_ attach-agent $node_(25) $sink25
$ns_ attach-agent $node_(26) $sink26
$ns_ attach-agent $node_(27) $sink27
$ns_ attach-agent $node_(28) $sink28
$ns_ attach-agent $node_(29) $sink29
$ns_ attach-agent $node_(30) $sink30
$ns_ attach-agent $node_(31) $sink31
$ns_ attach-agent $node_(32) $sink32
$ns_ attach-agent $node_(33) $sink33
$ns_ attach-agent $node_(34) $sink34
$ns_ attach-agent $node_(35) $sink35
$ns_ attach-agent $node_(36) $sink36
$ns_ attach-agent $node_(37) $sink37
$ns_ attach-agent $node_(38) $sink38
$ns_ attach-agent $node_(39) $sink39
$ns_ attach-agent $node_(40) $sink40
$ns_ attach-agent $node_(41) $sink41
$ns_ attach-agent $node_(42) $sink42


$ns_ attach-agent $node_(43) $sink43
$ns_ attach-agent $node_(44) $sink44
$ns_ attach-agent $node_(45) $sink45
$ns_ attach-agent $node_(46) $sink46
```

```
$ns_ attach-agent $node_(47) $sink47
$ns_ attach-agent $node_(48) $sink48
$ns_ attach-agent $node_(49) $sink49

  proc attach-CBtraffic { node sink } {
  #Get an instance of the simulator
  set ns_ [Simulator instance]
  #Create a CBR  agent and attach it to the node
  set udp [new Agent/UDP]
  $ns_ attach-agent $node $udp
  set cbr [new Application/Traffic/CBR]
  $cbr attach-agent $udp
  $cbr set packetSize_ 256          ;#sub packet size
  $cbr set interval_ 0.048

   #Attach CBR source to sink;
   $ns_ connect $udp $sink
   return $cbr
   }
   proc attach-CBR-traffic { node sink } {
   #Get an instance of the simulator
   set ns_ [Simulator instance]
   set udp [new Agent/UDP]
   $ns_ attach-agent $node $udp
   #Create a CBR  agent and attach it to the node
   set cbr [new Application/Traffic/CBR]
   $cbr attach-agent $udp
   $cbr set packetSize_ 256          ;#sub packet size
   $cbr set interval_ 0.048
   $cbr set random_ 1
   $cbr set maxpkts_ 5000

    #Attach CBR source to sink;
    $ns_ connect $udp $sink
    return $cbr
    }
   # ~~~~~~~~~~~~~~~~~~~~~~~~~~For route discovery~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

   set init [attach-CBR-traffic $node_(0) $sink21]

   set init1 [attach-CBR-traffic $node_(0) $sink21]
   set init2 [attach-CBR-traffic $node_(0) $sink44]




     $ns_ at 3.0 "$init start"
     $ns_ at 3.01 "$init stop"
     $ns_ at 37.8 "$init1 start"
```

```
$ns_ at 37.801 "$init1 stop"
$ns_ at 35.8 "$init2 start"
$ns_ at 35.801 "$init2 stop"


set dis [open E-Distance.txt w]
puts $dis
"\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
puts $dis "\tsource-Node\tDest-Node\tSX-cor\tSY-Cor\tE-Distance(d)"
puts $dis
"\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~"
close $dis
set nbr [open Neighbor w]
puts $nbr
"\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~"
puts $nbr "\tsource-Node\tNeighbor-Node\tH-Distance(d)"
puts $nbr
"\t~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
~~~~~~~~~~~~~~~~~~"
close $nbr
#~~~~~~~~~~~~~~~~~ For Calculation of Euclidean distance~~~~~~~~~~~~~~~~~~~~~~~~~~~

proc distance { n1 n2 nd1 nd2 fl} {
global r
set dis [open E-Distance.txt a]
set nbr [open Neighbor a]
set x1 [expr int([$n1 set X_])]
set y1 [expr int([$n1 set Y_])]
set x2 [expr int([$n2 set X_])]
set y2 [expr int([$n2 set Y_])]
set d [expr int(sqrt(pow(($x2-$x1),2)+pow(($y2-$y1),2)))]
if {$nd2>=$nd1} {
if {$fl == 49} {
puts $dis "\t$nd1\t\t$nd2\t\t$x1\t$y1\t$d"
}
}
if {$d<250} {
if {$nd2!=$nd1} {
puts $nbr "\t$nd1\t\t$nd2\t\t$d"
}
}
close $dis
close $nbr
}


#~~~~~~~~~~~~~~~~~~~~~~~~~~For Calculating the Residual Energy~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
proc energy {stnode etnode stime etime} {
set etp [open etmp w]
puts $etp "$stnode $etnode $stime $etime"
```

```
close $etp
exec awk -f energy.awk etmp Energy-Qos.tr
}
# ~~~~~~~~~~~~~~~~~~~~ For Energy based Routing ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

proc routing { tme stnode etnode snk qlen s_thres lc beta stme etme} {
set rtmp [open rtmp w]
puts $rtmp "$tme $stnode $etnode $snk $qlen $s_thres $lc $beta $stme $etme"
close $rtmp
exec awk -f routing.awk rtmp Res_ene($tme).txt E-Distance.txt
set rt [open route.txt r]
set rout [read $rt]
puts "$rout"
close $rt
}
# ~~~~~~~~~~~~~~~~~~~~~For function Calling ~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

for {set i 0} {$i<=49} {incr i} {
for {set j 0} {$j<=49} {incr j} {
$ns_ at 3.5 "distance $node_($i) $node_($j) $i $j 49"
}
}
$ns_ at 4.51 "energy 0 49 0 4.5"
$ns_ at 36.0 "energy 0 49 5 35.0"
$ns_ at 4.6 "routing 4.5 0 48 49 $qlen $s_thres $lc $beta 5.0 35.0"
$ns_ at 36.6 "routing 35.0 0 48 49 $qlen $s_thres $lc $beta 40.0 70.0"

set src Trans.tcl
$ns_ at 4.7 "source $src"
$ns_ at 36.7 "source $src"

# ~~~~~~~~~~~~~~~~~~~~~~~~~~Procedure for Energy calculation~~~~~~~~~~~~~~~~~~~~~~~

proc fenergy { timee } {
global ns_ t i sink0
set i [expr $i+5]
set time 5
set now [$ns_ now]
for {set i 0} {$i <=49} {incr i} {
set tmp [open temp1.tr w]




puts $tmp "[expr $timee-5] $timee $i"
close $tmp
exec awk -f renergy.awk temp1.tr Energy-Qos.tr
}
set tme [expr $now+$time]
$ns_ at $tme "energy $tme"
```

```
}
set i 0
set t 4
# For energy of each node

proc nenergy { t } {
global ns_
set cnt 0
set tot 0
set tm [$ns_ now]
set int 5
set rt [open route.txt r]
while {!([eof $rt])} {
set hp [gets $rt]
set tmp [open temp1.tr w]
puts $tmp "$t $hp"
close $tmp
set cnt [expr $cnt+1]
exec awk -f nenergy.awk temp1.tr Energy-Qos.tr
set tmp2 [open temp2.tr r]
set e [gets $tmp2]
set tene($cnt) $e
close $tmp2
set en [open ene($hp).tr a]
puts $en "$t\t$e"
close $en
puts "Node - $hp\tTime - $t\tEnergy - $e"
}
for {set k 1} {$k<=$cnt} {incr k} {
set tot [expr $tene($k)+$tot]
}
set avg [expr $tot/$cnt]
set pe_ene [open P-PriResEnergy.xg a]
set se_ene [open P-SecResEnergy.xg a]
if {$tm <=35} {
puts $pe_ene "$t $avg"
} elseif {$tm >=40 && $tm<=70 }  {
puts $se_ene "$t $avg"
}
close $pe_ene
close $se_ene
set tim [expr int($tm+$int)]
if {$tim <=70 } {




$ns_ at $tim "nenergy $tim"
}
}
# For Delay Calculation
```

```
set pdely [open pridelay w]
set sdely [open secdelay w]
close $pdely
close $sdely

#set tm 35

proc delay { start end send rec id} {
global ns_ dely rno
set tm $start
set t [open tdly.tr w]
puts $t "$start $end $send $rec"
close $t
exec awk -f delay.awk tdly.tr Energy-Qos.tr
set pdely [open pridelay a]
set sdely [open secdelay a]
set dly [open tmp$send r]
set value [gets $dly]
if {$tm<=30 && $id==1} {
puts $pdely "$value"
} elseif {$tm>=30 && $tm<=60 && $id==2} {
puts $sdely "$value"
}
close $pdely
close $sdely
set tm [expr $tm+5]
if {$tm<=30 && $id==1 } {
delay $tm [expr $tm+5] $send $rec $id
} elseif {$tm>=40 && $tm<70 && $id==2} {
delay $tm [expr $tm+5] $send $rec $id
} else {
return
}
}
set ptp [open P-PriThroughput.xg w]
set stp [open P-SecThroughput.xg w]
set pdp [open P-PriDrop.xg w]
set sdp [open P-SecDrop.xg w]
set ppdr [open P-PriPDR.xg w]
set spdr [open P-SecPDR.xg w]
puts $spdr "35 0"
set pdly [open P-PriDelay.xg w]
set sdly [open P-SecDelay.xg w]
set pe_ene [open P-PriResEnergy.xg w]
set se_ene [open P-SecResEnergy.xg w]
close $pe_ene


close $se_ene
close $pdly
close $sdly
proc record { } {
```

```
global ns_ sink0 sink14 sink28 sink15 sink16 sink24 sink33 sink31 sink26 sink37 sink46 sink49 ptp pdp ppdr
set t [$ns_ now]
set itval 5.0
set r0 [$sink0 set npkts_]
set r14 [$sink14 set npkts_]
set r28 [$sink28 set npkts_]
set r15 [$sink15 set npkts_]
set r16 [$sink16 set npkts_]


set r24 [$sink24 set npkts_]
set r33 [$sink33 set npkts_]
set r31 [$sink31 set npkts_]
set r26 [$sink26 set npkts_]
set r37 [$sink37 set npkts_]
set r46 [$sink46 set npkts_]
set r49 [$sink49 set npkts_]
set rec [expr ($r0+$r14+$r28+$r15+$r16+$r24+$r33+$r31+$r26+$r37+$r46+$r49)/12]
set l0 [$sink0 set nlost_]
set l14 [$sink14 set nlost_]
set l28 [$sink28 set nlost_]
set l15 [$sink15 set nlost_]
set l16 [$sink16 set nlost_]
set l24 [$sink24 set nlost_]
set l33 [$sink33 set nlost_]
set l31 [$sink31 set nlost_]
set l26 [$sink26 set nlost_]
set l37 [$sink37 set nlost_]
set l46 [$sink46 set nlost_]
set l49 [$sink49 set nlost_]
set los [expr ($l0+$l14+$l28+$l15+$l16+$l24+$l33+$l31+$l26+$l37+$l46+$l49)/12]
set b0 [$sink0 set bytes_]
set b14 [$sink14 set bytes_]
set b28 [$sink28 set bytes_]
set b15 [$sink15 set bytes_]
set b16 [$sink16 set bytes_]
set b24 [$sink24 set bytes_]
set b33 [$sink33 set bytes_]
set b31 [$sink31 set bytes_]
set b26 [$sink26 set bytes_]
set b37 [$sink37 set bytes_]


set b46 [$sink46 set bytes_]
set b49 [$sink49 set bytes_]
set byt [expr ($b0+$b14+$b28+$b15+$b16+$b24+$b33+$b31+$b26+$b37+$b46+$b49)/12]
set pdr 0
if {$rec!=0} {
set pdr [expr ($rec+0.0)/($rec+$los)]
}
set tput [expr ($byt*8.0)/($itval*1000000)]
puts $ptp "$t\t$tput"
puts $pdp "$t\t$los"
puts $ppdr "$t\t$pdr"
set inter [expr $t+5.0]
if {$inter <=35} {
$ns_ at $inter "record"
}
$sink0 set bytes_ 0
$sink14 set bytes_ 0
$sink28 set bytes_ 0
$sink15 set bytes_ 0
$sink16 set bytes_ 0
$sink24 set bytes_ 0
$sink33 set bytes_ 0
$sink31 set bytes_ 0
$sink26 set bytes_ 0
$sink37 set bytes_ 0
$sink46 set bytes_ 0
$sink49 set bytes_ 0
}
proc record1 { } {
global ns_ sink0 sink21 sink17 sink20 sink35 sink29 sink36 sink44 sink13 sink41 sink49 stp sdp spdr
set t [$ns_ now]
set itval 5.0
set r0 [$sink0 set npkts_]
set r21 [$sink21 set npkts_]
set r17 [$sink17 set npkts_]
set r20 [$sink20 set npkts_]
set r35 [$sink35 set npkts_]
set r29 [$sink29 set npkts_]
set r36 [$sink36 set npkts_]
set r44 [$sink44 set npkts_]
set r13 [$sink13 set npkts_]
set r41 [$sink41 set npkts_]
set r49 [$sink49 set npkts_]

set rec [expr ($r0+$r21+$r17+$r20+$r35+$r29+$r36+$r44+$r13+$r41+$r49)/11]

set l0 [$sink0 set nlost_]
set l21 [$sink21 set nlost_]
set l17 [$sink17 set nlost_]
set l20 [$sink20 set nlost_]


set l35 [$sink35 set nlost_]
set l29 [$sink29 set nlost_]
set l36 [$sink36 set nlost_]
set l44 [$sink44 set nlost_]
set l13 [$sink13 set nlost_]
set l41 [$sink41 set nlost_]
set l49 [$sink49 set nlost_]
set los [expr ($l0+$l21+$l17+$l20+$l35+$l29+$l36+$l44+$l13+$l41+$l49)/11]

set b0 [$sink0 set bytes_]
set b21 [$sink21 set bytes_]
set b17 [$sink17 set bytes_]
set b20 [$sink20 set bytes_]
set b35 [$sink35 set bytes_]
set b29 [$sink29 set bytes_]
set b36 [$sink36 set bytes_]


set b44 [$sink44 set bytes_]
set b13 [$sink13 set bytes_]
set b41 [$sink41 set bytes_]
set b49 [$sink49 set bytes_]
set byt [expr ($b0+$b21+$b17+$b20+$b35+$b29+$b36+$b44+$b13+$b41+$b49)/11]
set pdr 0
if {$rec!=0} {
set pdr [expr ($rec+0.0)/($rec+$los)]
}
set tput [expr ($byt*8.0)/(2*$itval*1000000)]
puts $stp "$t\t$tput"
puts $sdp "$t\t$los"
puts $spdr "$t\t$pdr"
set inter [expr $t+5.0]
if {$inter >=40 && $inter <= 70} {
$ns_ at $inter "record1"
}
$sink0 set bytes_ 0
$sink21 set bytes_ 0
$sink17 set bytes_ 0
$sink20 set bytes_ 0
$sink35 set bytes_ 0
$sink29 set bytes_ 0
$sink36 set bytes_ 0
$sink44 set bytes_ 0
$sink13 set bytes_ 0
$sink41 set bytes_ 0
$sink49 set bytes_ 0

}
$ns_ at 5.0 "record"
$ns_ at 40.0 "record1"
$ns_ at 5.0 "nenergy 5"


# Finish Procedure to exec NAM Window

proc finish {} {
        global ns_ namtrace ptp pdp ppdr stp sdp spdr
        $ns_ flush-trace
    close $namtrace
```

```
        close $ptp
        close $pdp
        close $ppdr
        close $stp
        close $sdp
        close $spdr
        exec awk -f adelay.awk pridelay
        exec awk -f adelay.awk secdelay
exec nam -r 5m Energy-Qos.nam &
        exec xgraph P-PriDelay.xg -geometry 800x400 -t "Primary Path Delay" -x "Time" -y "Avg Delay" -ly
0,0.07 &
        exec xgraph P-SecDelay.xg -geometry 800x400 -t "Secondary Path Delay" -x "Time" -y "Avg Delay"
-ly 0,0.07 &
      exec xgraph P-PriThroughput.xg -geometry 800x400 -t "Primary Path Throughput" -x "Time" -y "Mb/s" -
ly 0,0.1 &
        exec xgraph P-SecThroughput.xg -geometry 800x400 -t "Secondary Path Throughput" -x "Time" -y
"Mb/s" -ly 0,0.1 &
      exec xgraph P-PriDrop.xg -geometry 800x400 -t "Primary Path Packet Drop" -x "Time" -y "Avg Drop" &
      exec xgraph P-SecDrop.xg -geometry 800x400 -t "Secondary Path Packet Drop" -x "Time" -y "Avg
Drop" &
        exec xgraph P-PriResEnergy.xg -t "Primary Path Residual Energy in the Network" -x "Time" -y
"Energy" -ly 50,100 &
        exec xgraph P-SecResEnergy.xg -t "Secondary Path Residual Energy in the Network" -x "Time" -y
"Energy" -ly 0,70 &
        exec xgraph P-PriPDR.xg -geometry 800x400 -t "Primary Path PDR" -x "Time" -y "PDR" -ly 0,1 &
    exec xgraph P-SecPDR.xg -geometry 800x400 -t "Secondary Path PDR" -x "Time" -y "PDR" -ly 0.5,1 &
        #exec xgraph pridelay.tr -geometry 800x400 -t "Delay" -x "Time" -y "Avg Delay" &
        #exec xgraph secdelay.tr -geometry 800x400 -t "Delay" -x "Time" -y "Avg Delay" &
    #exec xgraph tput.tr -geometry 800x400 -t "Throughput" -x "Time" -y "Avg Throughput" &
    #exec xgraph drop.tr -geometry 800x400 -t "packet Drop" -x "Time" -y "Avg Drop" &
        #exec xgraph ene(0).tr ene(2).tr ene(16).tr ene(43).tr ene(48).tr ene(42).tr ene(49).tr &
        #exec xgraph e_energy.tr -t "Residual Energy in the Network" -x "Time" -y "Avg Residual Energy" &
        exit 0
        }

$ns_ at 101.0 "finish"


puts "Start of simulation.."
$ns_ run
```

Setdest.tr file

```
$ns_ at 1.00000000000 "$node_(0) setdest 10.114464158148 942.013027255123 2700.035066006541"
$ns_ at 1.00000000000 "$node_(1) setdest 563.691525046676 501.767963877818 1500.905881786818"
$ns_ at 1.00000000000 "$node_(2) setdest 815.846925787665 812.162515859540 4000.510692408826"
$ns_ at 1.00000000000 "$node_(3) setdest 949.375272298711 761.441029142526 990.811121758705"
$ns_ at 1.00000000000 "$node_(4) setdest 257.359213803596 96.220147495575 2500.777097114277"
$ns_ at 1.00000000000 "$node_(5) setdest 1307.223696334876 855.060762497745 2400.965615426853"
$ns_ at 1.00000000000 "$node_(6) setdest 225.225880583511 312.021221429206 2200.544625506104"
$ns_ at 1.00000000000 "$node_(7) setdest 351.996194853263 754.343957747226 2500.391012798373"
```

```
$ns_ at 1.00000000000 "$node_(8) setdest 506.204518193039 173.194032574156 2000.598308743082"
$ns_ at 1.00000000000 "$node_(9) setdest 637.524155210011 933.036542529147 2000.161486241871"
$ns_ at 1.00000000000 "$node_(10) setdest 758.090738537459 493.528115016839 1300.918261203216"
$ns_ at 1.00000000000 "$node_(11) setdest 108.391744840304 034.696071226332 1200.132044763645"
$ns_ at 1.00000000000 "$node_(12) setdest 928.901119627858 382.973138186711 1800.822636381981"
$ns_ at 1.00000000000 "$node_(13) setdest 1377.413553229072 382.661385758214 2000.959877918860"
$ns_ at 1.00000000000 "$node_(14) setdest 28.859140797216 739.086348976465 2700.358595449614"
$ns_ at 1.00000000000 "$node_(15) setdest 269.938168848664 504.571712680840 2600.600310735606"
$ns_ at 1.00000000000 "$node_(16) setdest 400.886480278426 496.818870206905 2700.409430227979"
$ns_ at 1.00000000000 "$node_(17) setdest 425.022206531982 906.390711583492 1300.395724330635"
$ns_ at 1.00000000000 "$node_(18) setdest 203.364052774320 719.687297091577 2600.485383787416"
$ns_ at 1.00000000000 "$node_(19) setdest 109.124617962614 184.138610053070 2400.206418999223"
$ns_ at 1.00000000000 "$node_(20) setdest 587.943723027663 806.072552142453 7000.648507312613"
$ns_ at 1.00000000000 "$node_(21) setdest 196.972319330611 912.626162021012 900.727807998048"
$ns_ at 1.00000000000 "$node_(22) setdest 071.632152343981 302.151511249951 2000.955992793396"


$ns_ at 1.00000000000 "$node_(29) setdest 910.148667711514 611.451410107755 1000.409298810906"
$ns_ at 1.00000000000 "$node_(30) setdest 354.631123501639 223.977622720261 1500.419918253865"
$ns_ at 1.00000000000 "$node_(31) setdest 756.740442836194 41.595608283739 1400.001516318488"
$ns_ at 1.00000000000 "$node_(32) setdest 1152.478912857271 699.488481317089 2400.607651650065"
$ns_ at 1.00000000000 "$node_(33) setdest 638.959989254288 165.029777173464 2600.939512808672"
$ns_ at 1.00000000000 "$node_(34) setdest 1055.638470005431 234.259756513412 1800.824120907909"
$ns_ at 1.00000000000 "$node_(35) setdest 746.906579999386 699.895116994666 1300.610131014350"
$ns_ at 1.00000000000 "$node_(36) setdest 1093.458789292590 548.551380225152 800.037439749061"
$ns_ at 1.00000000000 "$node_(37) setdest 1040.709567147212 65.081598471145 2100.641336675128"
$ns_ at 1.00000000000 "$node_(38) setdest 1110.656611247122 903.145032178184 2100.557110432507"
$ns_ at 1.00000000000 "$node_(39) setdest 945.445064397662 915.312115269927 1800.432124458226"
$ns_ at 1.00000000000 "$node_(40) setdest 1156.284324593642 145.153485914675 2800.140252633759"
$ns_ at 1.00000000000 "$node_(41) setdest 1376.869004383859 213.778480532096 1900.404170079172"
$ns_ at 1.00000000000 "$node_(42) setdest 1294.311390478343 144.339749226968 2300.560269228308"
$ns_ at 1.00000000000 "$node_(43) setdest 743.240246090566 318.589053081936 1500.470332407353"
$ns_ at 1.00000000000 "$node_(44) setdest 1268.388623240598 506.171293402936 1600.022746281869"
$ns_ at 1.00000000000 "$node_(45) setdest 1271.286282139725 302.914483394058 1600.571497474069"
$ns_ at 1.00000000000 "$node_(46) setdest 1253.033356101005 36.583208531446 1700.099295066822"
$ns_ at 1.00000000000 "$node_(47) setdest 477.613332179276 37.511546002329 2000.508512732519"


$ns_ at 1.00000000000 "$node_(48) setdest 1163.338793097639 380.288220334786 2500.723099738919"
$ns_ at 1.00000000000 "$node_(49) setdest 797.579663600905 60.584441336236 3000.777028599200"
$ns_ at 1.00000000000 "$node_(49) setdest 1406.739743458542 36.951211115777 2500.887179280290"

# creating Label

$ns_ at 3.0 "$node_(0) label Sink"
$ns_ at 3.41 "$node_(0) add-mark c2 blue4 hexagon"
$ns_ at 3.41 "$node_(0) color red1"

$ns_ at 3.0 "$node_(49) label Source"
$ns_ at 3.41 "$node_(49) add-mark c2 red4 circle"
```

```
$ns_ at 3.41 "$node_(49) color blue"

for {set i 1} {$i<49} {incr i} {
$ns_ at 3.4 "$node_($i) color purple"
$ns_ at 34.4 "$node_($i) color purple"
}
```

Routing.awk

```
BEGIN {
nd=-1
i=0
j=0
maxen=-1
maxe=0
m=1
}
{
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Temp File~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~


if(FILENAME=="rtmp") {
time=$1
stnode=$2
etnode=$3
snk=$4
buffer=$5
s_thres=$6
lc=$7
beta=$8
stme=$9
etme=$10

l=1
}
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ DISTANCE calculation~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

if(FILENAME=="E-Distance.txt") {
if($1>=0 && $1<=50) {
if(nd!=$1) {
flg=0
}
if(flg==0) {
nd=$1
flg=1


s[i,1]=$1
s[i,2]=$3
s[i,3]=$4
i++
```

```
}
}
}
#~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~ Residual Energy~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
if(FILENAME!="rtmp" && FILENAME!="Energy-Qos.tr" && FILENAME!="E-Distance.txt") {
if($2>=0 && $2<=50) {
e[j,1]=$2
e[j,2]=$3
j++
}
}
}
END {
tme=time+5
print "Routing at time - "time > "route.txt"
print 0 > "route.txt"
print
"*************************************************************************************************************************************" > "Routing-Table("time").txt"
print "\tSource\t\tIntermediate Nodes\t\tDest\tHcount\tTxm_Ene\t\tRes_Ene\t   Free_Buffer SS_Thres LC" >
"Routing-Table("time").txt"
print
"*************************************************************************************************************************************" > "Routing-Table("time").txt"
mene=0
tene=0
txene=0
xd=s[snk,2]
yd=s[snk,3]
for(sn=stnode;sn<=etnode;sn++) {
nxtnd=sn
inter=sn
while(nxtnd!=snk) {
inter=nxtnd
x1=s[nxtnd,2]
y1=s[nxtnd,3]
for(nn=0;nn<=49;nn++) {
x2=s[nn,2]
y2=s[nn,3]
d=sqrt(((x2-x1)^2)+((y2-y1)^2))
if(d<=250 && sn!=nn) {



if(nn==snk) {
mdis=d
mene=e[nn,2]
nxtnd=nn
tene=1+beta*d

} else {
```

```
if((x2>x1 && x1<1250 && y2<y1 && y1>100) || (y1<100 && x2>x1) || (x1>1250 && y2<y1)) {
cnod=nn
cdis=d
tene=1+beta*d
cene=e[nn,2]
if(mene<cene) {
mene=cene
mdis=cdis
cmnod=cnod
nxtnd=cmnod
tene=1+beta*mdis
}#max
}# Opt Neighbor
}
}# not in range
}# ifor

txene=txene+tene

if(sn==0) {
print nxtnd > "route.txt"
}
int_nodes[m]=inter
m++
cnode=-1
mene=0
mdis=0
cmnod=0
} #While
if(m==2) {

print "\t"int_nodes[1]"\t\t-\t\t\t\t"snk"\t"m-2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc > "Routing-
Table("time").txt"
} else if(m==3) {
print "\t"int_nodes[1]"\t\t"int_nodes[2]"\t\t\t\t"snk"\t"m-2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc
> "Routing-Table("time").txt"
} else if(m==4) {
print "\t"int_nodes[1]"\t\t"int_nodes[2]"-"int_nodes[3]"\t\t\t\t"snk"\t"m-
2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc > "Routing-Table("time").txt"



2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc > "Routing-Table("time").txt"
} else if(m==6) {
print "\t"int_nodes[1]"\t\t"int_nodes[2]"-"int_nodes[3]"-"int_nodes[4]"-"int_nodes[5]"\t\t\t"snk"\t"m-
2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc > "Routing-Table("time").txt"
} else if(m==7) {
print "\t"int_nodes[1]"\t\t"int_nodes[2]"-"int_nodes[3]"-"int_nodes[4]"-"int_nodes[5]"-
"int_nodes[6]"\t\t\t"snk"\t"m-2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc > "Routing-
Table("time").txt"
} else if(m==8) {
```

```
print "\t"int_nodes[1]"\t\t"int_nodes[2]"-"int_nodes[3]"-"int_nodes[4]"-"int_nodes[5]"-"int_nodes[6]"-
"int_nodes[7]"\t\t"snk"\t"m-2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc > "Routing-
Table("time").txt"
} else if(m==9) {
print "\t"int_nodes[1]"\t\t"int_nodes[2]"-"int_nodes[3]"-"int_nodes[4]"-"int_nodes[5]"-"int_nodes[6]"-
"int_nodes[7]"-"int_nodes[8]"\t\t"snk"\t"m-2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc > "Routing-
Table("time").txt"
} else if(m==10) {
print "\t"int_nodes[1]"\t\t"int_nodes[2]"-"int_nodes[3]"-"int_nodes[4]"-"int_nodes[5]"-"int_nodes[6]"-
"int_nodes[7]"-"int_nodes[8]"-"int_nodes[9]"\t\t"snk"\t"m-
2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc > "Routing-Table("time").txt"
} else if(m==11) {
print "\t"int_nodes[1]"\t\t"int_nodes[2]"-"int_nodes[3]"-"int_nodes[4]"-"int_nodes[5]"-"int_nodes[6]"-
"int_nodes[7]"-"int_nodes[8]"-"int_nodes[9]"-"int_nodes[10]" \t"snk"\t"m-
2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc > "Routing-Table("time").txt"
if(int_nodes[1]==0) {
frac=0.00
for(ss=1;ss<=9;ss++) {
print "set cbr"int_nodes[ss]"_"int_nodes[ss+1]" [attach-CBR-traffic $node_("int_nodes[ss]")
$sink"int_nodes[ss+1]"]" > "Trans.tcl"
print "$ns_ at "stme+frac" \"$cbr"int_nodes[ss]"_"int_nodes[ss+1]" start\"" > "Trans.tcl"
print "$ns_ at "etme" \"$cbr"int_nodes[ss]"_"int_nodes[ss+1]" stop\"" > "Trans.tcl"
print "$ns_ at "tme+0.0" \"$node_("int_nodes[ss+1]") color chocolate\"" > "Trans.tcl"
print "$ns_ at "75.5+frac" \"delay "stme" "stme+5" "int_nodes[ss]" "int_nodes[ss+1]" 2\"" > "Trans.tcl"
frac=frac+0.01
}
print "set cbr"int_nodes[10]"_"snk" [attach-CBR-traffic $node_("int_nodes[10]") $sink"snk"]" > "Trans.tcl"



#print "$ns_ at "tme+0.0" \"$node_("") color blue\"" > "Trans.tcl"
print "$ns_ at "75.5+frac" \"delay "stme" "stme+5" "int_nodes[ss]" "snk" 2\"" > "Trans.tcl"
}

} else if(m==12) {
print "\t"int_nodes[1]"\t\t"int_nodes[2]"-"int_nodes[3]"-"int_nodes[4]"-"int_nodes[5]"-"int_nodes[6]"-
"int_nodes[7]"-"int_nodes[8]"-"int_nodes[9]"-"int_nodes[10]"-"int_nodes[11]"\t"snk"\t"m-
2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc > "Routing-Table("time").txt"
if(int_nodes[1]==0) {
frac=0.00
for(ss=1;ss<=10;ss++) {
print "set cbr"int_nodes[ss]"_"int_nodes[ss+1]" [attach-CBR-traffic $node_("int_nodes[ss]")
$sink"int_nodes[ss+1]"]" > "Trans.tcl"
print "$ns_ at "stme+frac" \"$cbr"int_nodes[ss]"_"int_nodes[ss+1]" start\"" > "Trans.tcl"
print "$ns_ at "etme" \"$cbr"int_nodes[ss]"_"int_nodes[ss+1]" stop\"" > "Trans.tcl"
print "$ns_ at "tme-4.0" \"$node_("int_nodes[ss+1]") color chocolate\"" > "Trans.tcl"

print "$ns_ at "40+frac" \"delay 5 10 "int_nodes[ss]" "int_nodes[ss+1]" 1\"" > "Trans.tcl"

frac=frac+0.01
}
print "set cbr"int_nodes[11]"_"snk" [attach-CBR-traffic $node_("int_nodes[11]") $sink"snk"]" > "Trans.tcl"
print "$ns_ at "stme+frac" \"$cbr"int_nodes[11]"_"snk" start\"" > "Trans.tcl"
print "$ns_ at "etme" \"$cbr"int_nodes[11]"_"snk" stop\"" > "Trans.tcl"
#print "$ns_ at "tme+0.0" \"$node_("") color blue\"" > "Trans.tcl"
print "$ns_ at "40+frac" \"delay 5 10 "int_nodes[ss]" "snk" 1\"" > "Trans.tcl"
}
} else if(m==13) {
print "\t"int_nodes[1]"\t\t"int_nodes[2]"-"int_nodes[3]"-"int_nodes[4]"-"int_nodes[5]"-"int_nodes[6]"-
"int_nodes[7]"-"int_nodes[8]"-"int_nodes[9]"-"int_nodes[10]"-"int_nodes[11]"-"int_nodes[12]"\t"snk"\t"m-
2"\t"txene/1000"\t\t"e[inter,2]"\t"buffer"\t"s_thres"\t"lc > "Routing-Table("time").txt"
}
txene=0
tene=0
m=1
} # for
}
```

Residual Energy.awk

```
BEGIN {
stnode=0
etnode=0
stime=0
etime=0
n=-1
t=0
nd=-1
ene=0
}
{
if(FILENAME == "etmp") {
stnode=$1
etnode=$2
stime=$3
etime=$4
}

if(FILENAME != "etmp") {
n=$1
t=$3
nd=$5
ene=$7
if($1 == "N") {
        for(i=stnode;i<=etnode;i++) {
                if(t>=stime && t <= etime) {
                                        if(nd==i) {

                                                node[i,1]=i
                                                node[i,2]=ene
                                                        }
                                                }
                                        }
                              }
                    }



}
END {
print "  ****************************************" > "Res_ene("etime").txt"
print "\tTime\tNode-id\tResidual-Energy" > "Res_ene("etime").txt"
print "  ****************************************" > "Res_ene("etime").txt"
for(i=stnode;i<=etnode;i++) {
print "\t"etime"\t"node[i,1]"\t"node[i,2] > "Res_ene("etime").txt"
}
}
```

Energy.awk

```
BEGIN {
nod=0
energy=0
time=0
node=0

}
{
if(FILENAME == "temp1.tr") {
node=$2
time=$1
}

if(FILENAME == "Energy-Qos.tr") {
n=$1
t=$3
nd=$5
ene=$7
if($1 == "N") {
          if(nd==node) {
                    if(t>time-5 && t <= time) {

nod=node
energy=ene
tme=t
}
}
}
}
}

END {
print energy > "temp2.tr";}
```

Primary Delay.awk

```
BEGIN {
i=1
}
{
if(FILENAME=="pridelay" || FILENAME=="secdelay") {
fle=FILENAME
d[i,1]=$1
d[i,2]=$2
i++
}


}
END {
if(fle=="pridelay") {
for(k=10;k<=35;k=k+5) {
for(j=1;j<i;j++) {
if(k==d[j,1]) {
t[k]=t[k]+d[j,2]
}
}
print k" "t[k] > "P-PriDelay.xg"
}
}
if(fle=="secdelay") {
for(k=45;k<=70;k=k+5) {
for(j=1;j<i;j++) {
if(k==d[j,1]) {
t[k]=t[k]+d[j,2]
}
}
print k" "t[k] > "P-SecDelay.xg"
}
}
}
```

Secondary delay.awk

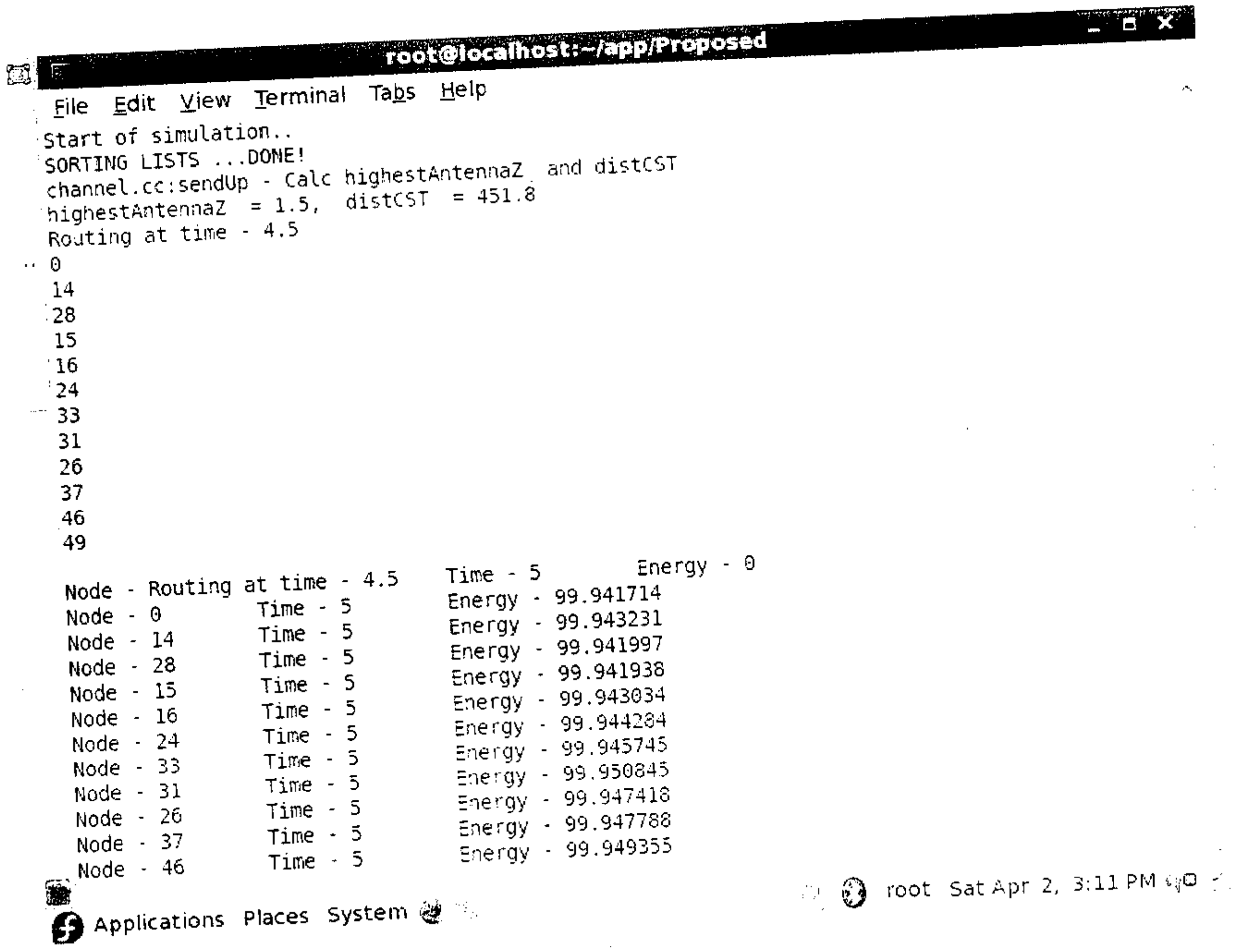
```
BEGIN {
st=0
et=0
i=1
f=0
stime=0
etime=0
```

```
seh=0
k=1
seq=0



tot=0
avg=0
flg=0

}
{
if(FILENAME=="tdly.tr") {
start=$1
end=$2
sender=$3
rec=$4
}


if(FILENAME=="Energy-Qos.tr") {
if($3>=start && $3 <=end) {
if($1=="s" || $1=="r" || $1=="f" || $1=="d") {
if($31~sender && $33~rec) {
if($7>=0) {
if(flg == 0) {
sta=$47

flg=1
}
else {
en=$47
}
val[i,1]=$3
val[i,2]=$47

i++
}
}
}
}
}
}
END {
seq=en-sta
for(j=sta;j<en;j++) {

                    while(k<=i) {
                                        if(j==val[k,2]) {
                                                        if(f == 0) {
                                                                f=1
                                                                stime=val[k,1]
```

```
                                                    }
                                                    else {
                                                    etime=val[k,1]
                                                    seh=val[k,2]
                                                          }
                                              }
                                k++
                                }

                          dlay=etime-stime
                          tot=tot+dlay
                          f=0
                          k=0
                          stime=0
                          etime=0
                          seh=0
                          } avg=tot/seq
          if(avg<0) {
          avg=0
```

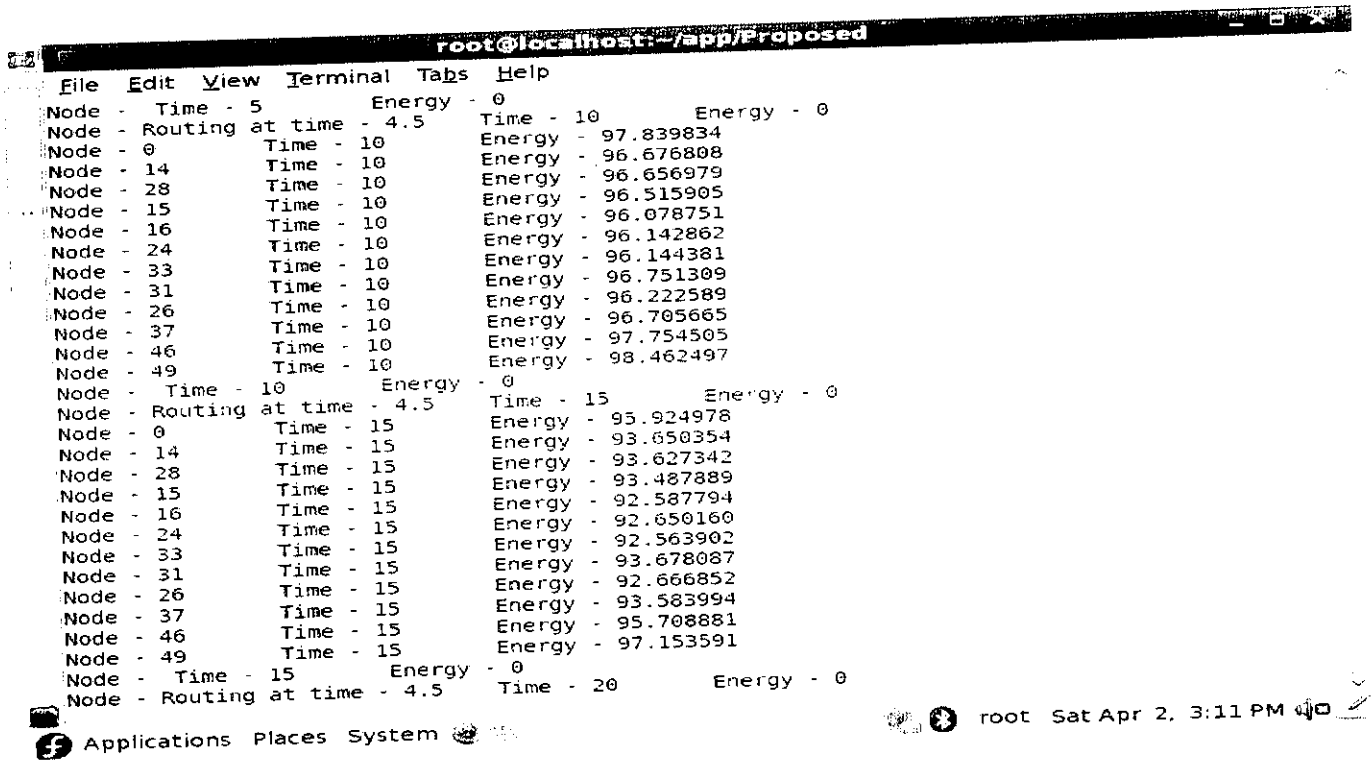
6.2 SNAP SHOTS

- **Routing Information**


root@localhost:~/app/Proposed
File Edit View Terminal Tabs Help
Applications Places System
root Sat Apr 2, 3:11 PM

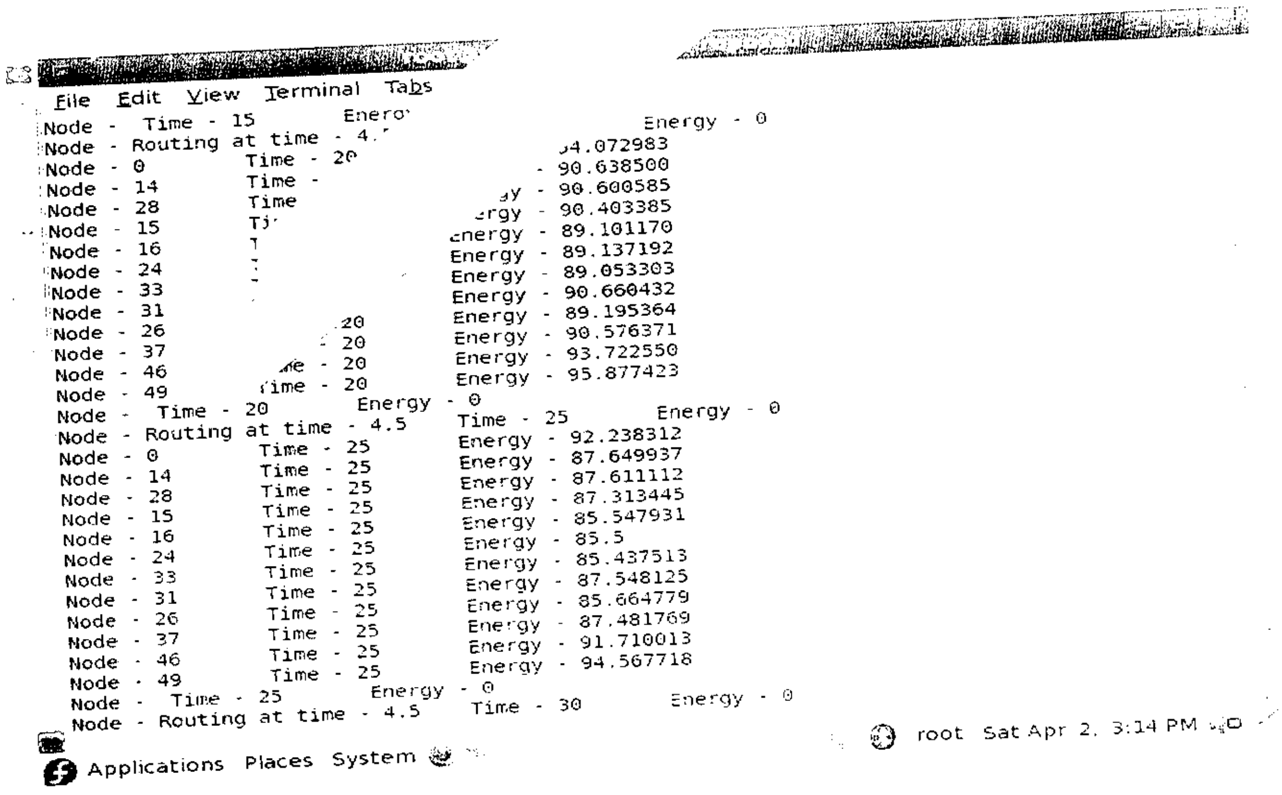
```
Start of simulation..
SORTING LISTS ...DONE!
channel.cc:sendUp - Calc highestAntennaZ and distCST
highestAntennaZ  = 1.5,  distCST  = 451.8
Routing at time - 4.5
0
14
28
15
16
24
33
31
26
37
46
49

Node - Routing at time - 4.5   Time - 5        Energy - 0
Node - 0        Time - 5       Energy - 99.941714
Node - 14       Time - 5       Energy - 99.943231
Node - 28       Time - 5       Energy - 99.941997
Node - 15       Time - 5       Energy - 99.941938
Node - 16       Time - 5       Energy - 99.943034
Node - 24       Time - 5       Energy - 99.944284
Node - 33       Time - 5       Energy - 99.945745
Node - 31       Time - 5       Energy - 99.950845
Node - 26       Time - 5       Energy - 99.947418
Node - 37       Time - 5       Energy - 99.947788
Node - 46       Time - 5       Energy - 99.949355
```

Here the routing information for the primary path is chosen at the Routing time of 4.5, and routing table is updated. It also lists the information of Node, Time at 5 and Energy.


root@localhost:~/app/Proposed
File Edit View Terminal Tabs Help
Applications Places System
root Sat Apr 2, 3:11 PM

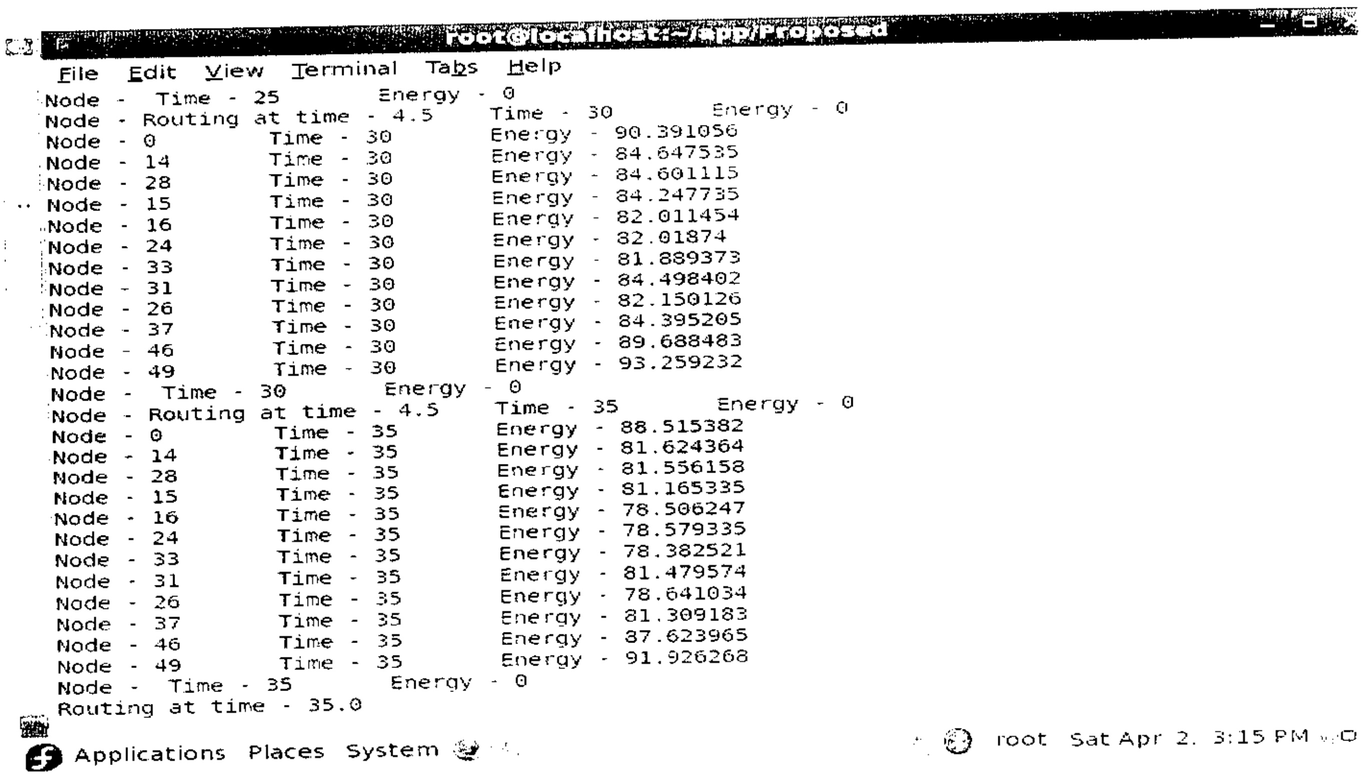
```
Node -  Time - 5         Energy - 0
Node - Routing at time - 4.5   Time - 10       Energy - 0
Node - 0          Time - 10    Energy - 97.839834
Node - 14         Time - 10    Energy - 96.676808
Node - 28         Time - 10    Energy - 96.656979
Node - 15         Time - 10    Energy - 96.515905
Node - 16         Time - 10    Energy - 96.078751
Node - 24         Time - 10    Energy - 96.142862
Node - 33         Time - 10    Energy - 96.144381
Node - 31         Time - 10    Energy - 96.751309
Node - 26         Time - 10    Energy - 96.222589
Node - 37         Time - 10    Energy - 96.705665
Node - 46         Time - 10    Energy - 97.754505
Node - 49         Time - 10    Energy - 98.462497
Node -  Time - 10        Energy - 0
Node - Routing at time - 4.5   Time - 15       Energy - 0
Node - 0          Time - 15    Energy - 95.924978
Node - 14         Time - 15    Energy - 93.650354
Node - 28         Time - 15    Energy - 93.627342
Node - 15         Time - 15    Energy - 93.487889
Node - 16         Time - 15    Energy - 92.587794
Node - 24         Time - 15    Energy - 92.650160
Node - 33         Time - 15    Energy - 92.563902
Node - 31         Time - 15    Energy - 93.678087
Node - 26         Time - 15    Energy - 92.666852
Node - 37         Time - 15    Energy - 93.583994
Node - 46         Time - 15    Energy - 95.708881
Node - 49         Time - 15    Energy - 97.153591
Node -  Time - 15        Energy - 0
Node - Routing at time - 4.5   Time - 20       Energy - 0
```

Here the routing information for the primary path is chosen at the Routing time of 4.5, and routing table is updated. It also lists the information of Node, Time at 10& 15 and Energy.



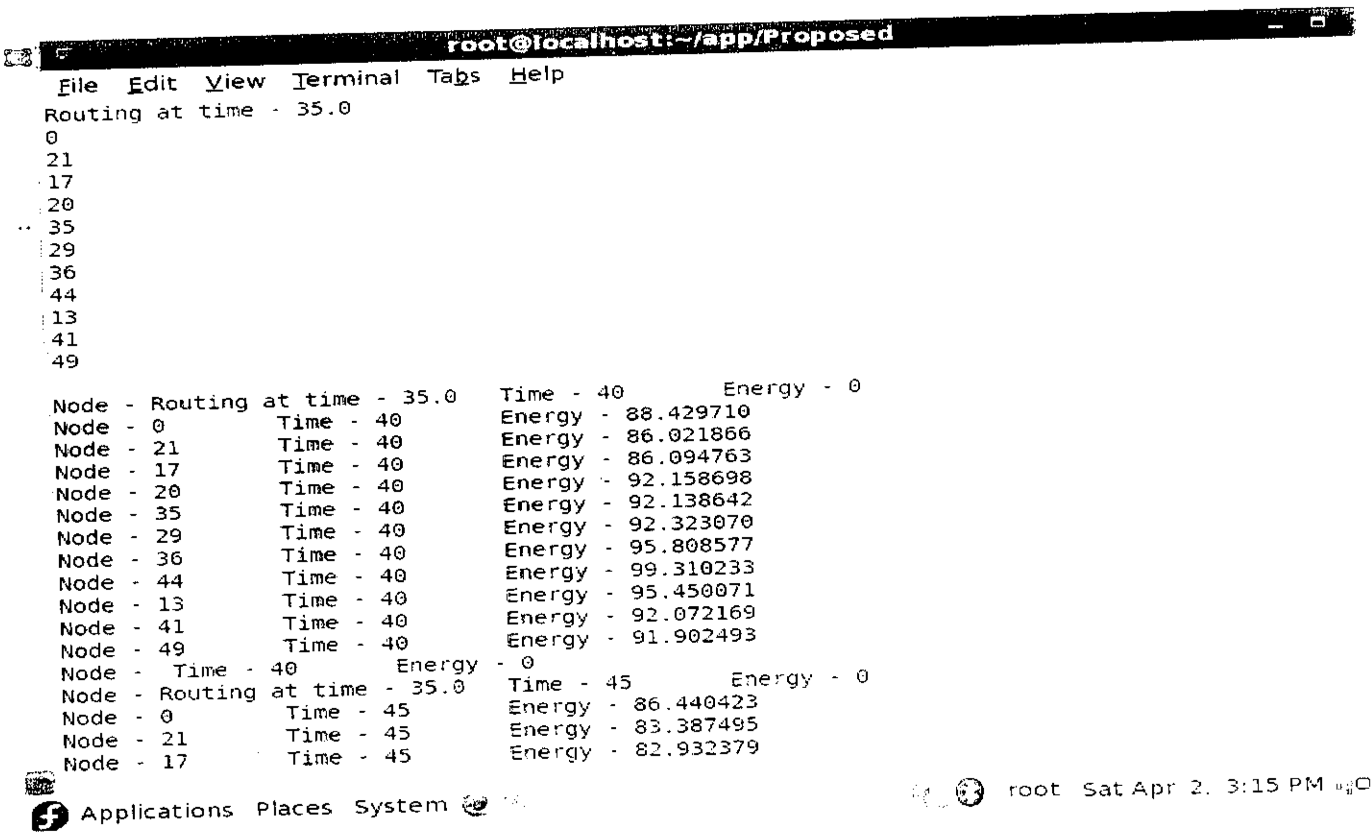
```
File  Edit  View  Terminal  Tabs
Node -  Time - 15        Enero                                 Energy - 0
Node - Routing at time - 4.                       4.072983
Node - 0          Time - 20                     - 90.638500
Node - 14         Time -                     y  - 90.600585
Node - 28         Time                    ergy  - 90.403385
Node - 15         Ti                     nergy  - 89.101170
Node - 16         T                     Energy  - 89.137192
Node - 24                               Energy  - 89.053303
Node - 33                               Energy  - 90.660432
Node - 31                               Energy  - 89.195364
Node - 26                  20           Energy  - 90.576371
Node - 37                  20           Energy  - 93.722550
Node - 46            e  -  20           Energy  - 95.877423
Node - 49          ime  -  20           Energy  - 95.877423
Node -  Time - 20         Energy - 0
Node - Routing at time - 4.5     Time - 25            Energy - 0
Node - 0          Time - 25     Energy - 92.238312
Node - 14         Time - 25     Energy - 87.649937
Node - 28         Time - 25     Energy - 87.611112
Node - 15         Time - 25     Energy - 87.313445
Node - 16         Time - 25     Energy - 85.547931
Node - 24         Time - 25     Energy - 85.5
Node - 33         Time - 25     Energy - 85.437513
Node - 31         Time - 25     Energy - 87.548125
Node - 26         Time - 25     Energy - 85.664779
Node - 37         Time - 25     Energy - 87.481769
Node - 46         Time - 25     Energy - 91.710013
Node - 49         Time - 25     Energy - 94.567718
Node -  Time - 25         Energy - 0
Node - Routing at time - 4.5     Time - 30            Energy - 0
```

Here the routing information for the primary path is chosen at the Routing time of 4.5, and routing table is updated. It also lists the information of Node, Time at 20 & 25 and Energy.


root@localhost:~/app/Proposed
File Edit View Terminal Tabs Help
Applications Places System
root Sat Apr 2. 3:15 PM

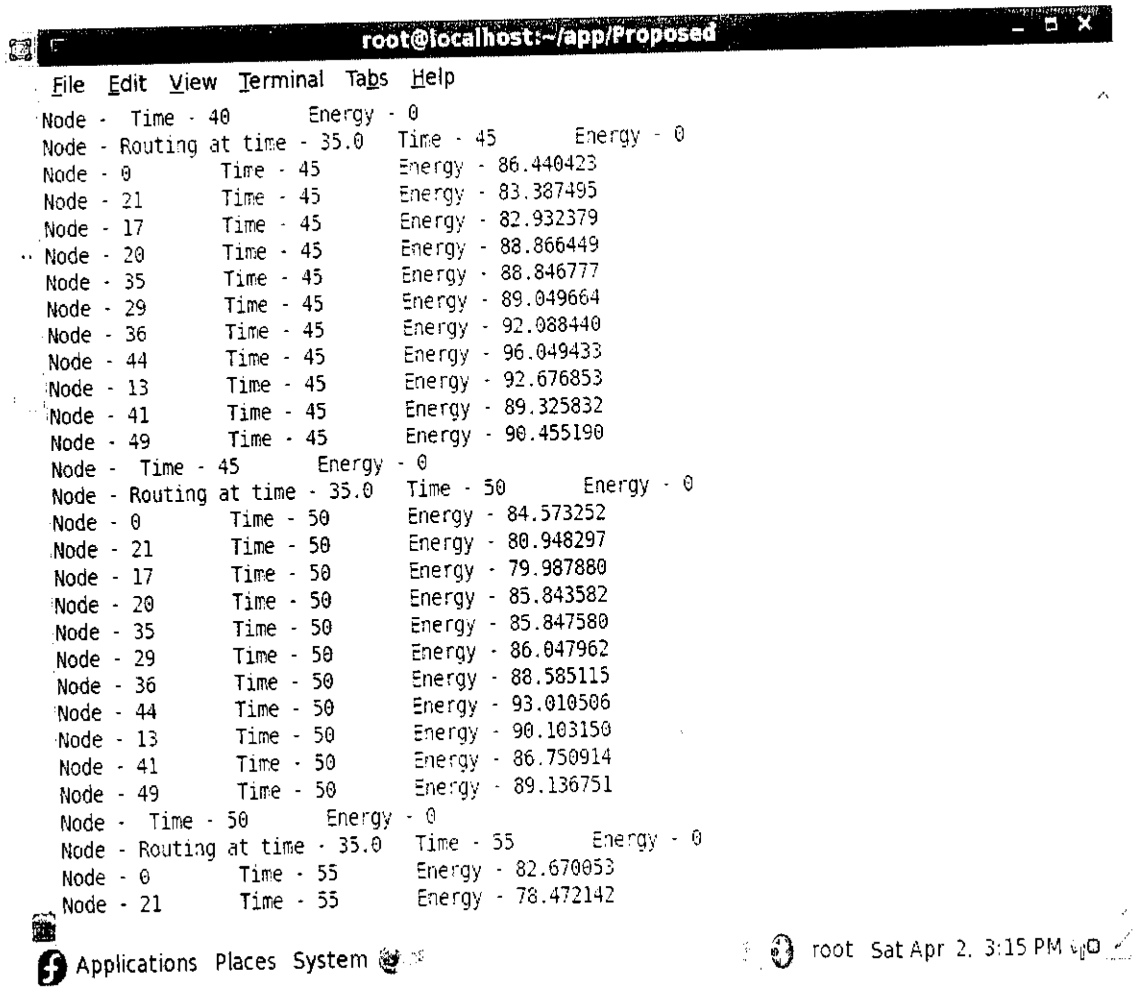
```
Node -   Time - 25        Energy - 0
Node - Routing at time - 4.5     Time - 30        Energy - 0
Node - 0          Time - 30      Energy - 90.391056
Node - 14         Time - 30      Energy - 84.647535
Node - 28         Time - 30      Energy - 84.601115
Node - 15         Time - 30      Energy - 84.247735
Node - 16         Time - 30      Energy - 82.011454
Node - 24         Time - 30      Energy - 82.01874
Node - 33         Time - 30      Energy - 81.889373
Node - 31         Time - 30      Energy - 84.498402
Node - 26         Time - 30      Energy - 82.150126
Node - 37         Time - 30      Energy - 84.395205
Node - 46         Time - 30      Energy - 89.688483
Node - 49         Time - 30      Energy - 93.259232
Node -   Time - 30        Energy - 0
Node - Routing at time - 4.5     Time - 35        Energy - 0
Node - 0          Time - 35      Energy - 88.515382
Node - 14         Time - 35      Energy - 81.624364
Node - 28         Time - 35      Energy - 81.556158
Node - 15         Time - 35      Energy - 81.165335
Node - 16         Time - 35      Energy - 78.506247
Node - 24         Time - 35      Energy - 78.579335
Node - 33         Time - 35      Energy - 78.382521
Node - 31         Time - 35      Energy - 81.479574
Node - 26         Time - 35      Energy - 78.641034
Node - 37         Time - 35      Energy - 81.309183
Node - 46         Time - 35      Energy - 87.623965
Node - 49         Time - 35      Energy - 91.926268
Node -   Time - 35        Energy - 0
Routing at time - 35.0
```

Here the routing information for the primary path is chosen at the Routing time of 4.5, and routing table is updated. It also lists the information of Node, Time at 30 1nd 35 and Energy.


root@localhost:~/app/Proposed
File Edit View Terminal Tabs Help
Applications Places System
root Sat Apr 2, 3:15 PM

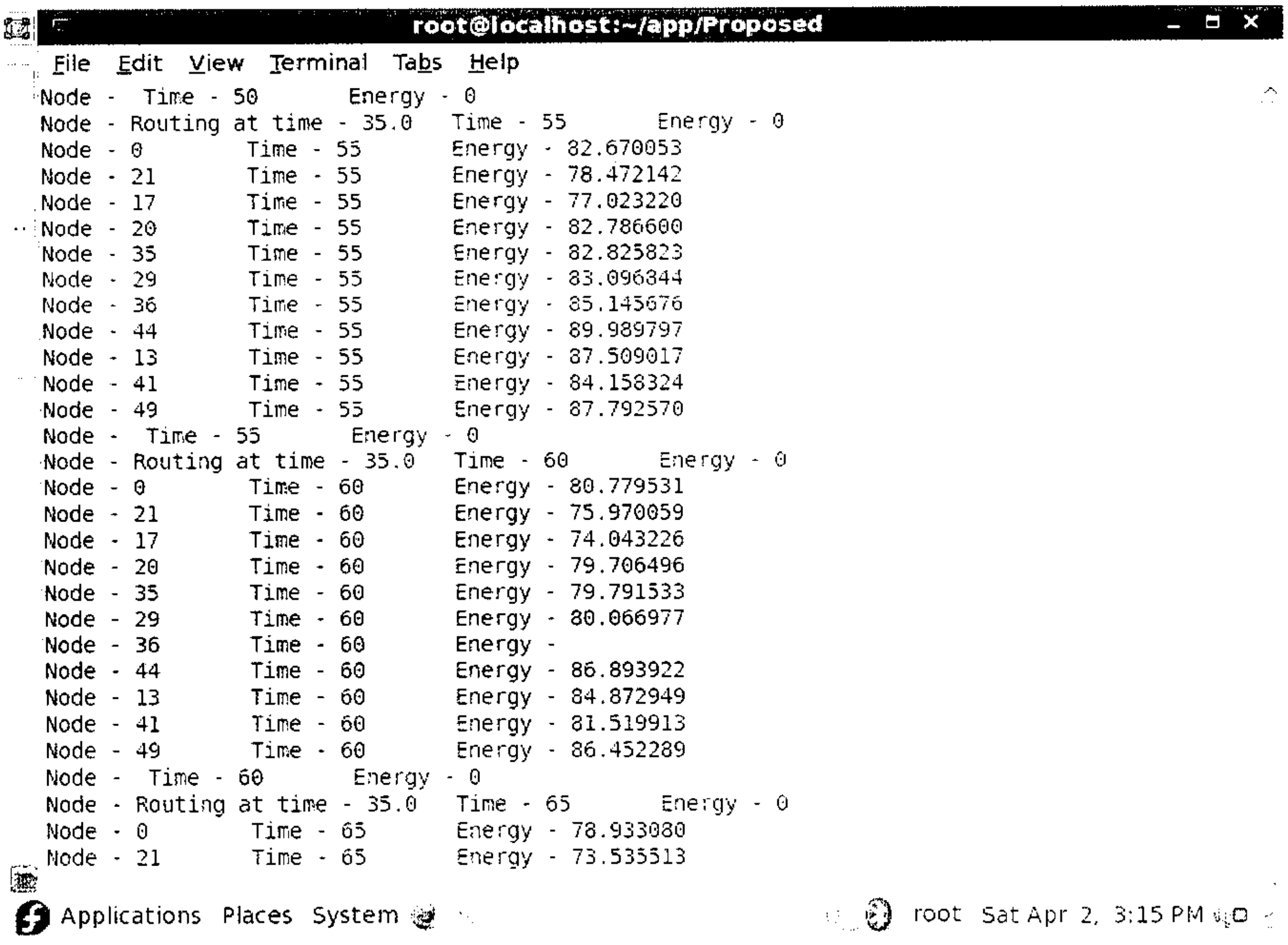
```
Routing at time - 35.0
0
21
17
20
35
29
36
44
13
41
49

Node - Routing at time - 35.0   Time - 40        Energy - 0
Node - 0          Time - 40     Energy - 88.429710
Node - 21         Time - 40     Energy - 86.021866
Node - 17         Time - 40     Energy - 86.094763
Node - 20         Time - 40     Energy - 92.158698
Node - 35         Time - 40     Energy - 92.138642
Node - 29         Time - 40     Energy - 92.323070
Node - 36         Time - 40     Energy - 95.808577
Node - 44         Time - 40     Energy - 99.310233
Node - 13         Time - 40     Energy - 95.450071
Node - 41         Time - 40     Energy - 92.072169
Node - 49         Time - 40     Energy - 91.902493
Node -   Time - 40         Energy - 0
Node - Routing at time - 35.0   Time - 45        Energy - 0
Node - 0          Time - 45     Energy - 86.440423
Node - 21         Time - 45     Energy - 83.387495
Node - 17         Time - 45     Energy - 82.932379
```

Here the routing information for the Secondary path is chosen at the Routing time of 35.0, and routing table is updated. It also lists the information of Node, Time at 40 & 45 and Energy.


root@localhost:~/app/Proposed
File Edit View Terminal Tabs Help
Applications Places System
root Sat Apr 2, 3:15 PM

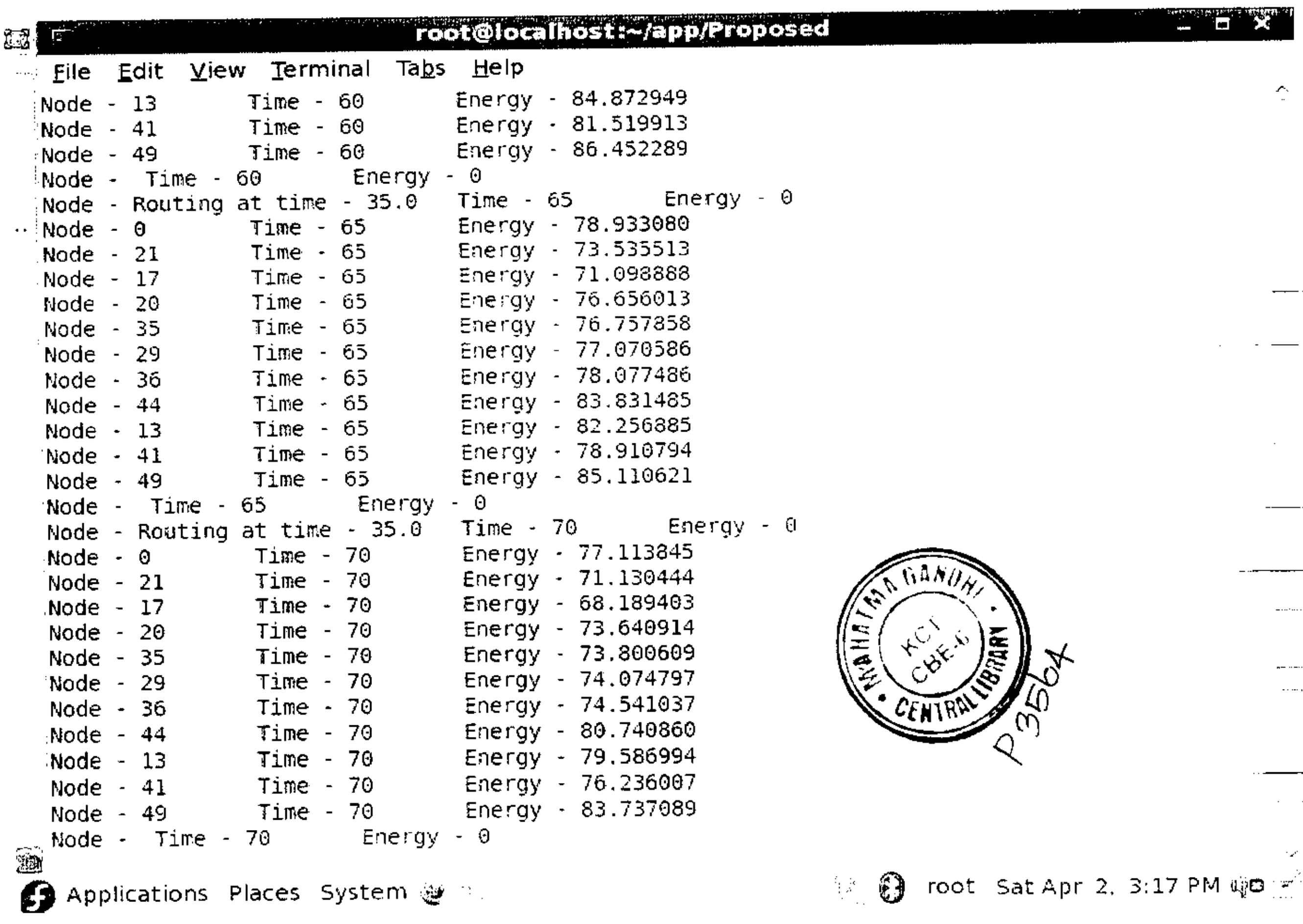
```
Node -  Time - 40       Energy - 0
Node - Routing at time - 35.0  Time - 45       Energy - 0
Node - 0        Time - 45       Energy - 86.440423
Node - 21       Time - 45       Energy - 83.387495
Node - 17       Time - 45       Energy - 82.932379
Node - 20       Time - 45       Energy - 88.866449
Node - 35       Time - 45       Energy - 88.846777
Node - 29       Time - 45       Energy - 89.049664
Node - 36       Time - 45       Energy - 92.088440
Node - 44       Time - 45       Energy - 96.049433
Node - 13       Time - 45       Energy - 92.676853
Node - 41       Time - 45       Energy - 89.325832
Node - 49       Time - 45       Energy - 90.455190
Node -  Time - 45       Energy - 0
Node - Routing at time - 35.0  Time - 50       Energy - 0
Node - 0        Time - 50       Energy - 84.573252
Node - 21       Time - 50       Energy - 80.948297
Node - 17       Time - 50       Energy - 79.987880
Node - 20       Time - 50       Energy - 85.843582
Node - 35       Time - 50       Energy - 85.847580
Node - 29       Time - 50       Energy - 86.047962
Node - 36       Time - 50       Energy - 88.585115
Node - 44       Time - 50       Energy - 93.010506
Node - 13       Time - 50       Energy - 90.103150
Node - 41       Time - 50       Energy - 86.750914
Node - 49       Time - 50       Energy - 89.136751
Node -  Time - 50       Energy - 0
Node - Routing at time - 35.0  Time - 55       Energy - 0
Node - 0        Time - 55       Energy - 82.670053
Node - 21       Time - 55       Energy - 78.472142
```

Here the routing information for the Secondary path is chosen at the Routing time of 35.0, and routing table is updated. It also lists the information of Node, Time at 45 & 50 and Energy.


root@localhost:~/app/Proposed
File Edit View Terminal Tabs Help
Applications Places System
root Sat Apr 2, 3:15 PM

```
Node -  Time - 50       Energy - 0
Node - Routing at time - 35.0   Time - 55       Energy - 0
Node - 0        Time - 55       Energy - 82.670053
Node - 21       Time - 55       Energy - 78.472142
Node - 17       Time - 55       Energy - 77.023220
Node - 20       Time - 55       Energy - 82.786600
Node - 35       Time - 55       Energy - 82.825823
Node - 29       Time - 55       Energy - 83.096344
Node - 36       Time - 55       Energy - 85.145676
Node - 44       Time - 55       Energy - 89.989797
Node - 13       Time - 55       Energy - 87.509017
Node - 41       Time - 55       Energy - 84.158324
Node - 49       Time - 55       Energy - 87.792570
Node -  Time - 55       Energy - 0
Node - Routing at time - 35.0   Time - 60       Energy - 0
Node - 0        Time - 60       Energy - 80.779531
Node - 21       Time - 60       Energy - 75.970059
Node - 17       Time - 60       Energy - 74.043226
Node - 20       Time - 60       Energy - 79.706496
Node - 35       Time - 60       Energy - 79.791533
Node - 29       Time - 60       Energy - 80.066977
Node - 36       Time - 60       Energy -
Node - 44       Time - 60       Energy - 86.893922
Node - 13       Time - 60       Energy - 84.872949
Node - 41       Time - 60       Energy - 81.519913
Node - 49       Time - 60       Energy - 86.452289
Node -  Time - 60       Energy - 0
Node - Routing at time - 35.0   Time - 65       Energy - 0
Node - 0        Time - 65       Energy - 78.933080
Node - 21       Time - 65       Energy - 73.535513
```

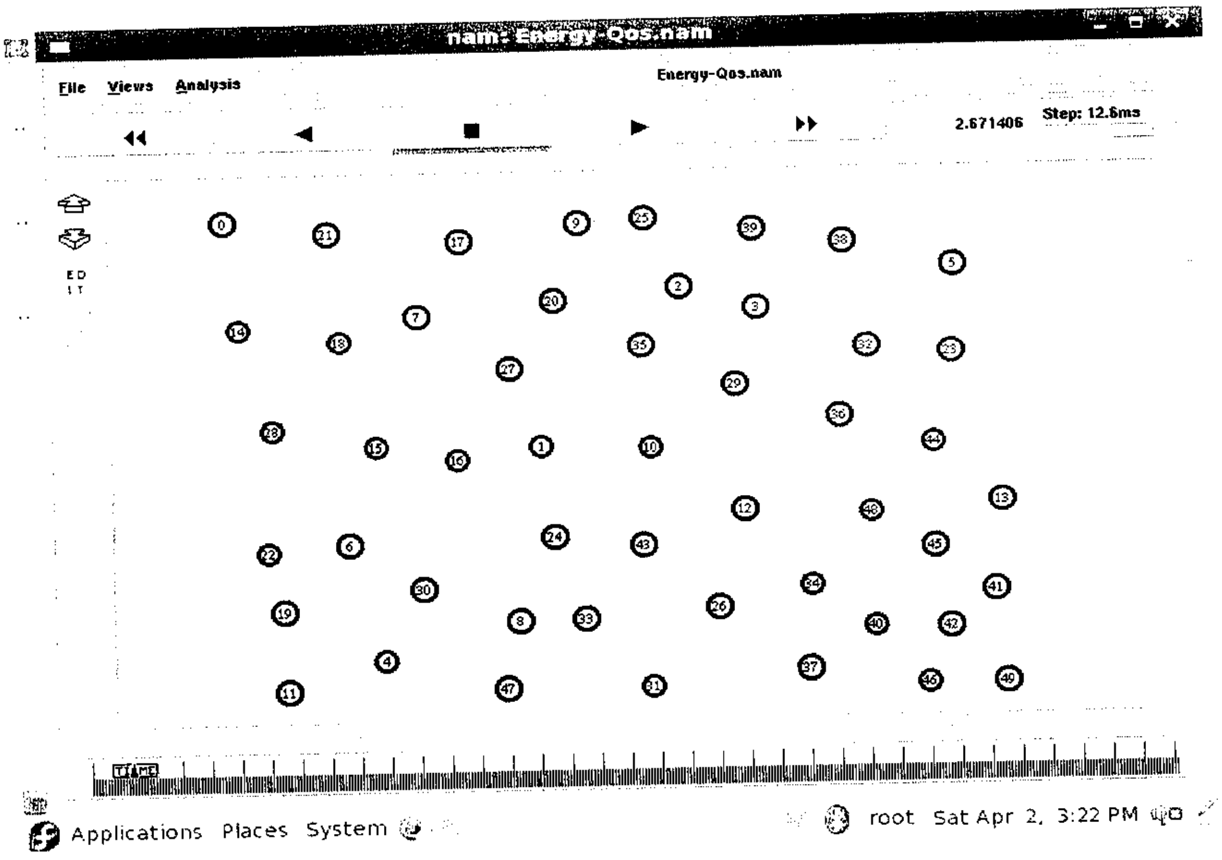
Here the routing information for the Secondary path is chosen at the Routing time of 35.0, and routing table is updated. It also lists the information of Node, Time at 55 & 60 and Energy.


root@localhost:~/app/Proposed
File Edit View Terminal Tabs Help
Applications Places System
root Sat Apr 2, 3:17 PM

```
Node - 13       Time - 60       Energy - 84.872949
Node - 41       Time - 60       Energy - 81.519913
Node - 49       Time - 60       Energy - 86.452289
Node -  Time - 60       Energy - 0
Node - Routing at time - 35.0   Time - 65       Energy - 0
Node - 0        Time - 65       Energy - 78.933080
Node - 21       Time - 65       Energy - 73.535513
Node - 17       Time - 65       Energy - 71.098888
Node - 20       Time - 65       Energy - 76.656013
Node - 35       Time - 65       Energy - 76.757858
Node - 29       Time - 65       Energy - 77.070586
Node - 36       Time - 65       Energy - 78.077486
Node - 44       Time - 65       Energy - 83.831485
Node - 13       Time - 65       Energy - 82.256885
Node - 41       Time - 65       Energy - 78.910794
Node - 49       Time - 65       Energy - 85.110621
Node -  Time - 65       Energy - 0
Node - Routing at time - 35.0   Time - 70       Energy - 0
Node - 0        Time - 70       Energy - 77.113845
Node - 21       Time - 70       Energy - 71.130444
Node - 17       Time - 70       Energy - 68.189403
Node - 20       Time - 70       Energy - 73.640914
Node - 35       Time - 70       Energy - 73.800609
Node - 29       Time - 70       Energy - 74.074797
Node - 36       Time - 70       Energy - 74.541037
Node - 44       Time - 70       Energy - 80.740860
Node - 13       Time - 70       Energy - 79.586994
Node - 41       Time - 70       Energy - 76.236007
Node - 49       Time - 70       Energy - 83.737089
Node -  Time - 70       Energy - 0
```

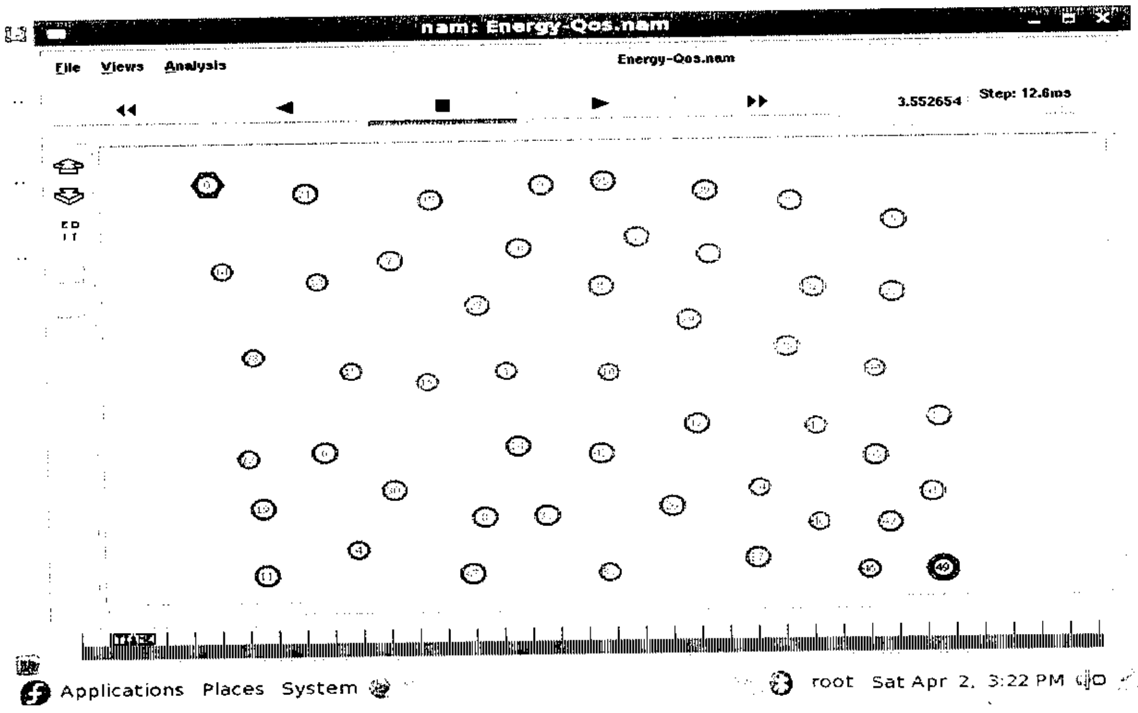
Here the routing information for the Secondary path is chosen at the Routing time of 35.0, and routing table is updated. It also lists the information of Node, Time at 65 & 70 and Energy.

- **Initializations of Nodes**

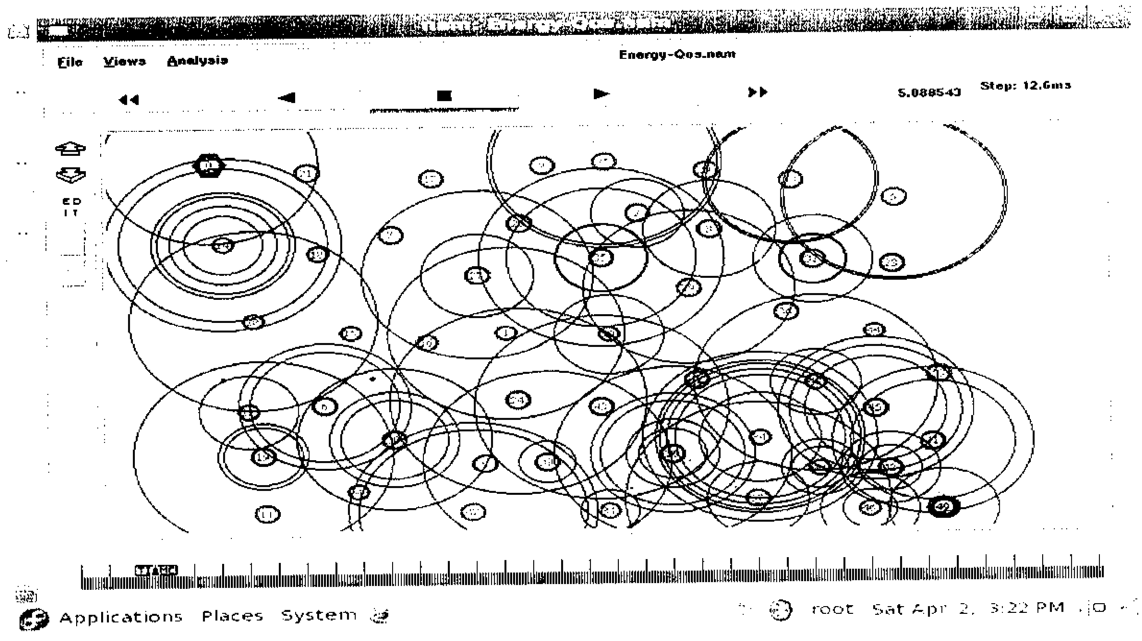


Here the Nodes are Deployed.

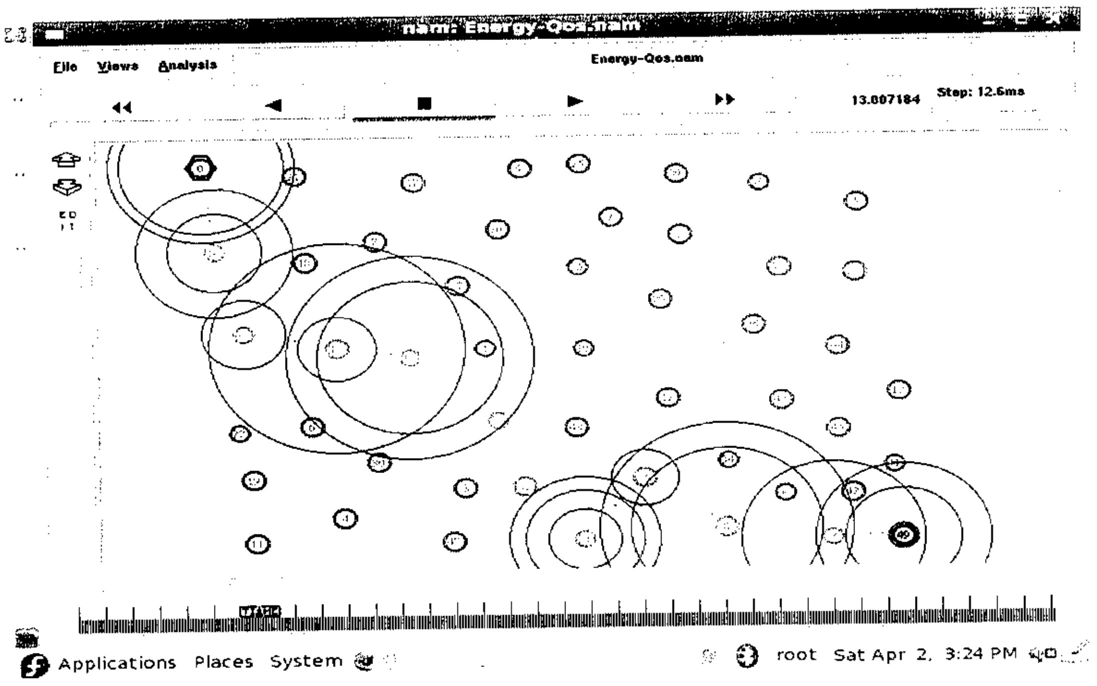
- **Initializations of Source and Sink**



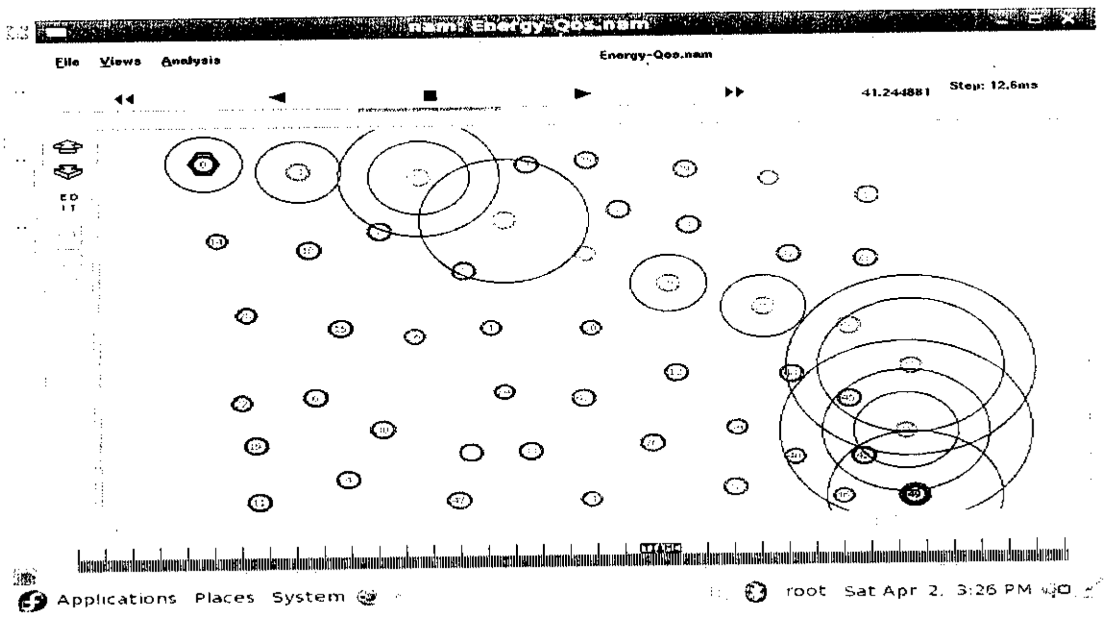
- **Energy Transmission**



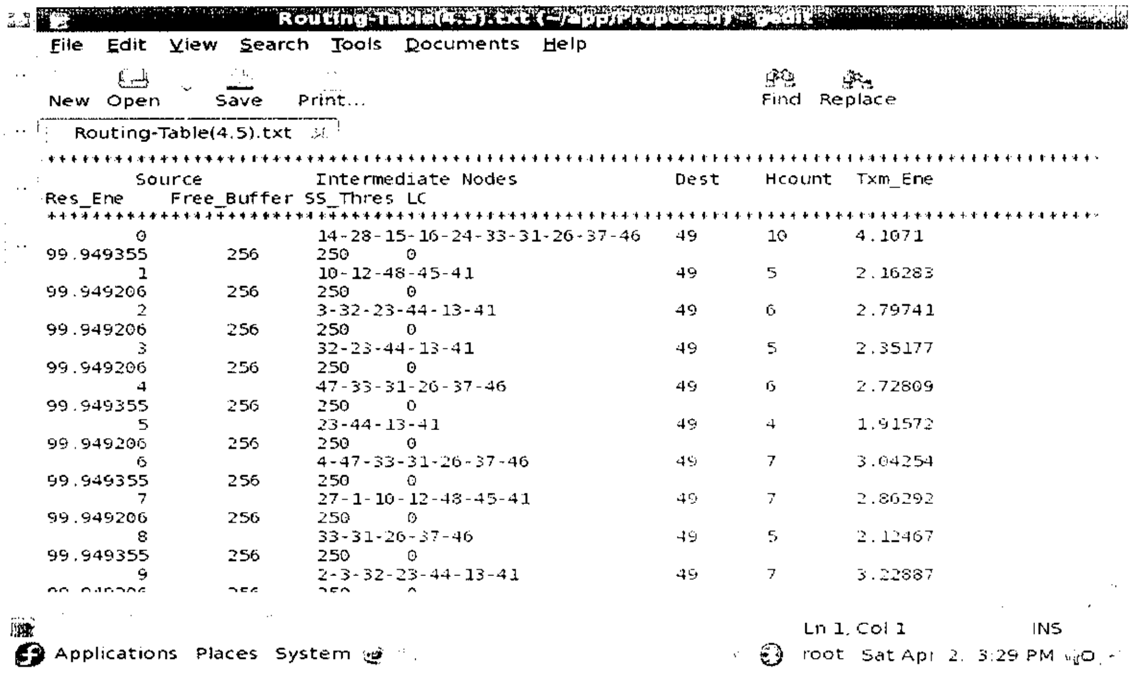
- **Primary Path**



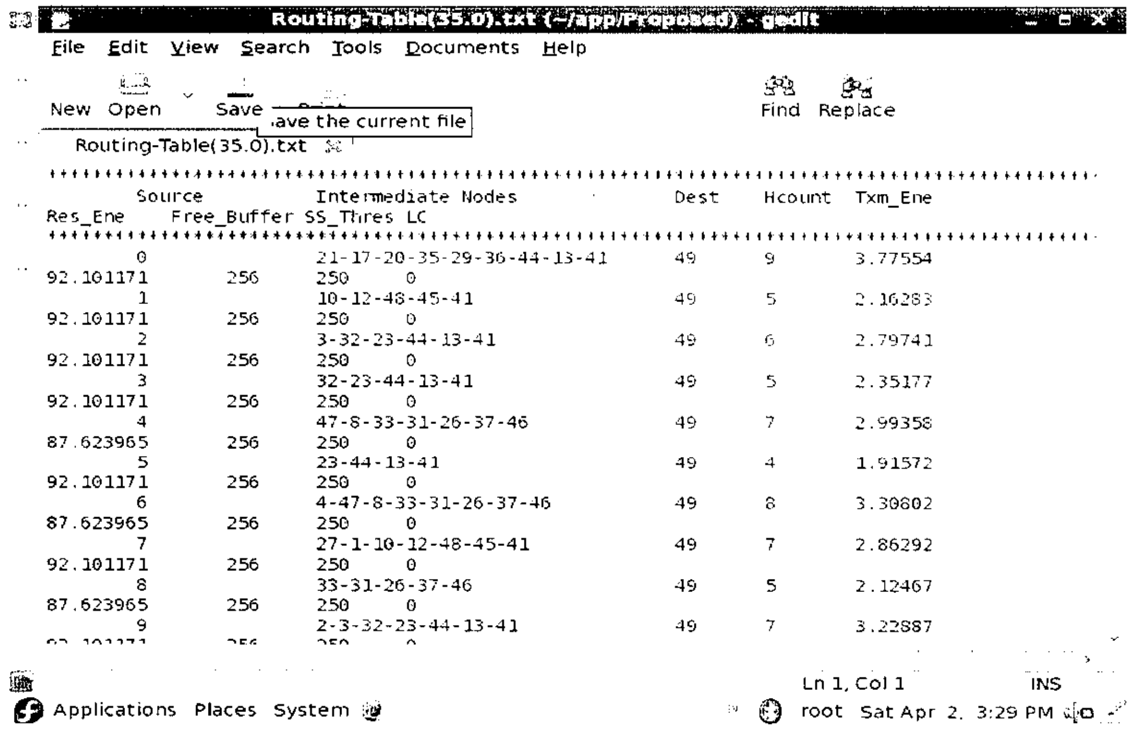
- **Secondary Path**



Routing Table

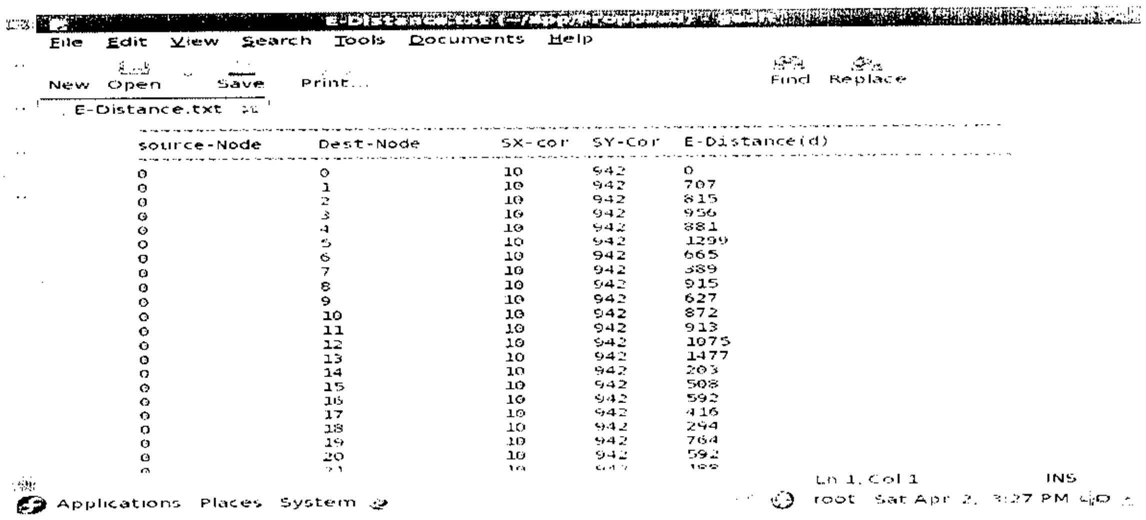

Routing-Table(4.5).txt (~/app/Proposed) - gedit
File Edit View Search Tools Documents Help
New Open Save Print...
Find Replace
Routing-Table(4.5).txt
Ln 1, Col 1 INS
Applications Places System
root Sat Apr 2, 3:29 PM

```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        Source            Intermediate Nodes                  Dest      Hcount   Txm_Ene
Res_Ene      Free_Buffer SS_Thres LC
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
          0               14-28-15-16-24-33-31-26-37-46      49        10       4.1071
99.949355       256       250       0
          1               10-12-48-45-41                     49        5        2.16283
99.949206       256       250       0
          2               3-32-23-44-13-41                   49        6        2.79741
99.949206       256       250       0
          3               32-23-44-13-41                     49        5        2.35177
99.949206       256       250       0
          4               47-33-31-26-37-46                  49        6        2.72809
99.949355       256       250       0
          5               23-44-13-41                        49        4        1.91572
99.949206       256       250       0
          6               4-47-33-31-26-37-46                49        7        3.04254
99.949355       256       250       0
          7               27-1-10-12-48-45-41                49        7        2.86292
99.949206       256       250       0
          8               33-31-26-37-46                     49        5        2.12467
99.949355       256       250       0
          9               2-3-32-23-44-13-41                 49        7        3.22887
```


Routing-Table(35.0).txt (~/app/Proposed) - gedit
File Edit View Search Tools Documents Help
New Open Save
ave the current file
Find Replace
Routing-Table(35.0).txt
Ln 1, Col 1 INS
Applications Places System
root Sat Apr 2, 3:29 PM

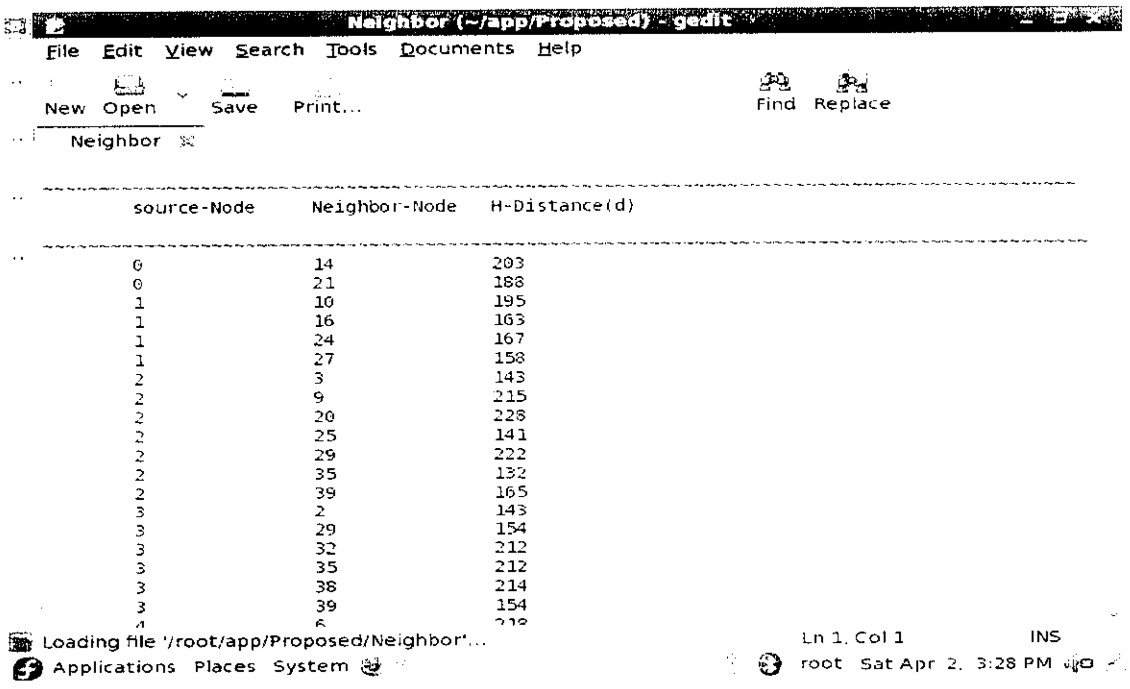
```
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
        Source            Intermediate Nodes                  Dest      Hcount   Txm_Ene
Res_Ene      Free_Buffer SS_Thres LC
++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++
          0               21-17-20-35-29-36-44-13-41         49        9        3.77554
92.101171       256       250       0
          1               10-12-48-45-41                     49        5        2.16283
92.101171       256       250       0
          2               3-32-23-44-13-41                   49        6        2.79741
92.101171       256       250       0
          3               32-23-44-13-41                     49        5        2.35177
92.101171       256       250       0
          4               47-8-33-31-26-37-46                49        7        2.99358
87.623965       256       250       0
          5               23-44-13-41                        49        4        1.91572
92.101171       256       250       0
          6               4-47-8-33-31-26-37-46              49        8        3.30802
87.623965       256       250       0
          7               27-1-10-12-48-45-41                49        7        2.86292
92.101171       256       250       0
          8               33-31-26-37-46                     49        5        2.12467
87.623965       256       250       0
          9               2-3-32-23-44-13-41                 49        7        3.22887
```

- **Distance Table**


E-Distance.txt (~/app/Proposed) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Find Replace
E-Distance.txt
Ln 1, Col 1 INS
Applications Places System
root Sat Apr 2, 3:27 PM

```
source-Node     Dest-Node        SX-cor  SY-Cor  E-Distance(d)
0               0                10      942     0
0               1                10      942     707
0               2                10      942     815
0               3                10      942     956
0               4                10      942     881
0               5                10      942     1299
0               6                10      942     665
0               7                10      942     389
0               8                10      942     915
0               9                10      942     627
0               10               10      942     872
0               11               10      942     913
0               12               10      942     1075
0               13               10      942     1477
0               14               10      942     203
0               15               10      942     508
0               16               10      942     592
0               17               10      942     416
0               18               10      942     294
0               19               10      942     764
0               20               10      942     592
```

- **Neighbor Table**

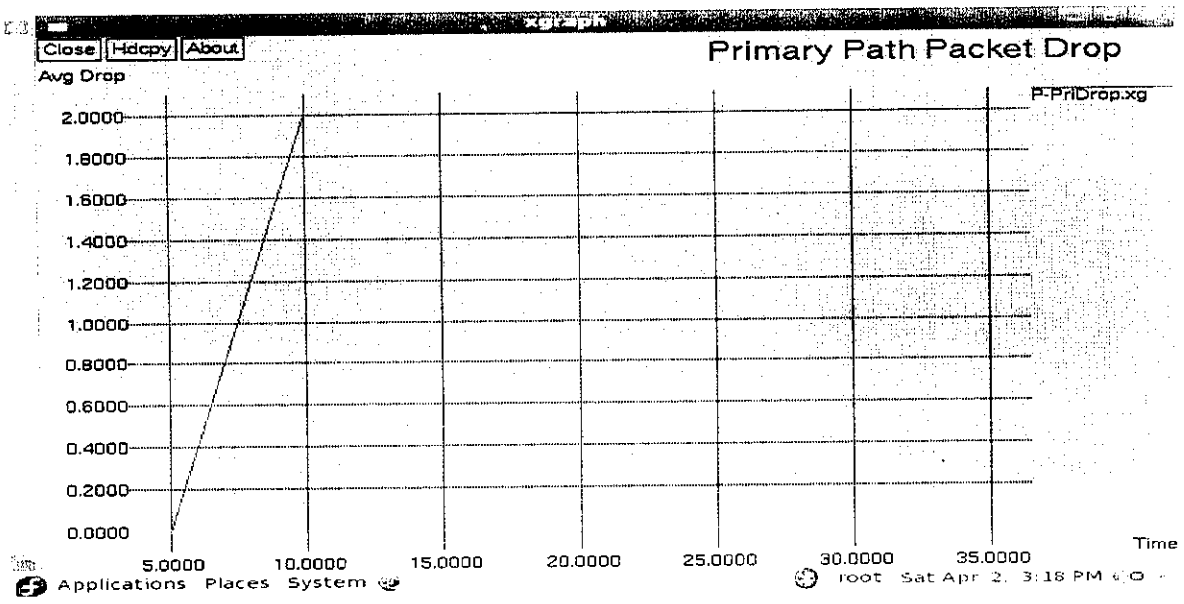

Neighbor (~/app/Proposed) - gedit
File Edit View Search Tools Documents Help
New Open Save Print... Find Replace
Neighbor
Ln 1, Col 1 INS
Applications Places System
root Sat Apr 2, 3:28 PM

```
source-Node     Neighbor-Node    H-Distance(d)

0               14               203
0               21               188
1               10               195
1               16               163
1               24               167
1               27               158
2               3                143
2               9                215
2               20               228
2               25               141
2               29               222
2               35               132
2               39               165
3               2                143
3               29               154
3               32               212
3               35               212
3               38               214
3               39               154
```

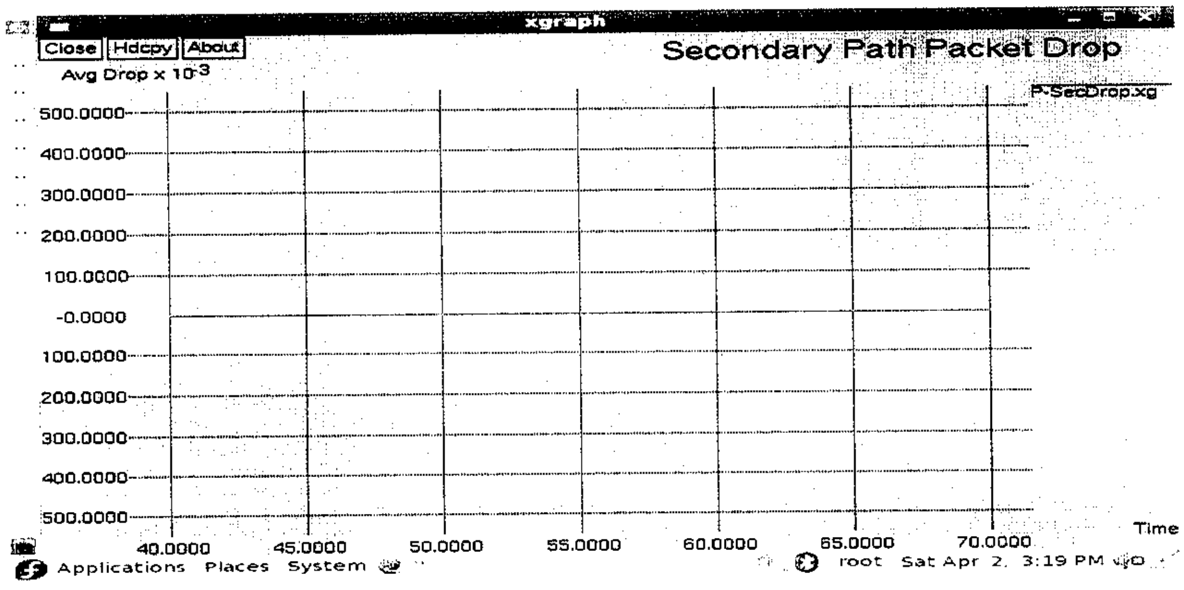
Loading file '/root/app/Proposed/Neighbor'...

Graph

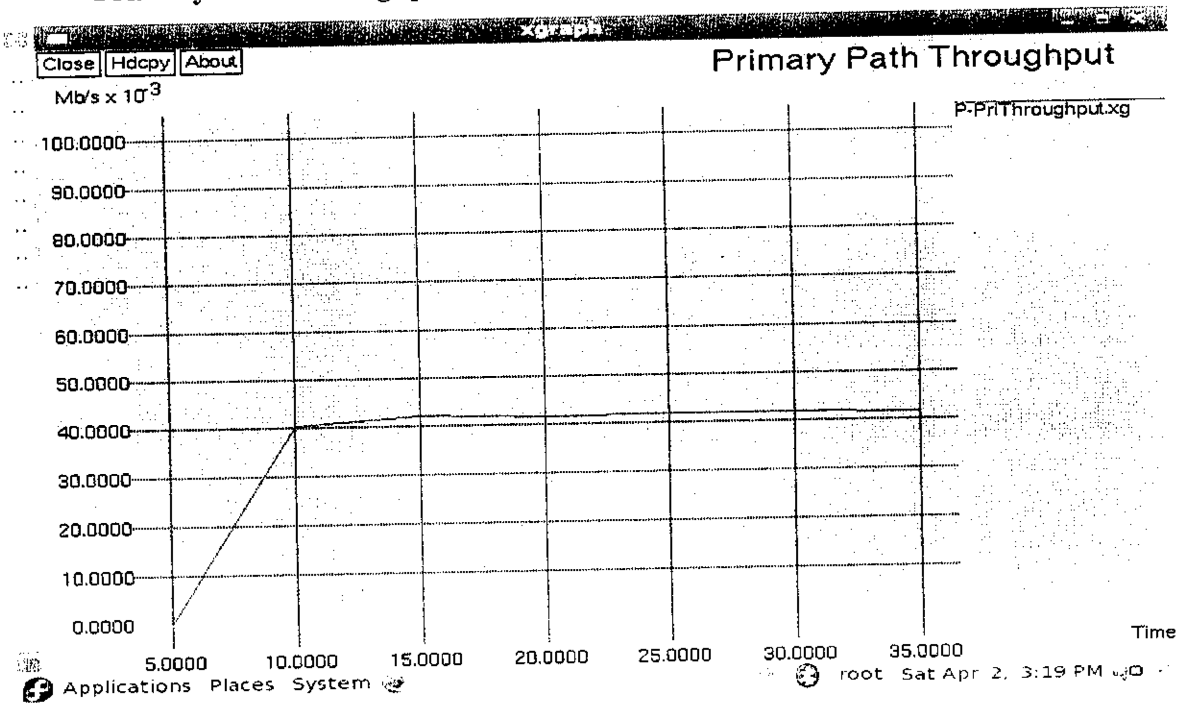
- **Primary Path Packet Drop**



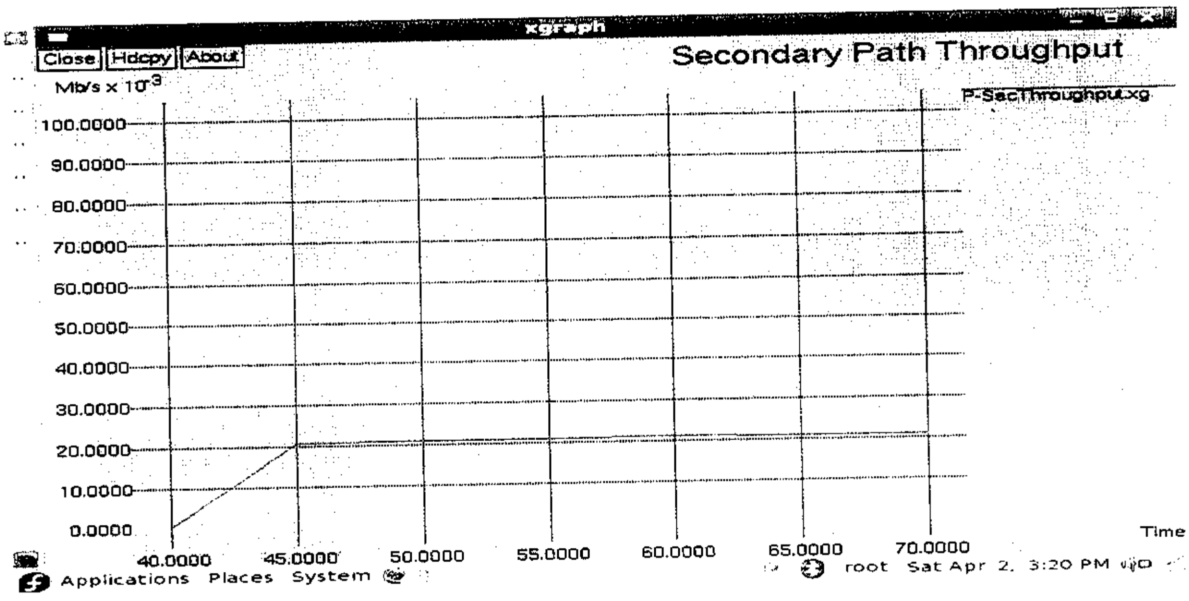
- **Secondary Path Packet Drop**



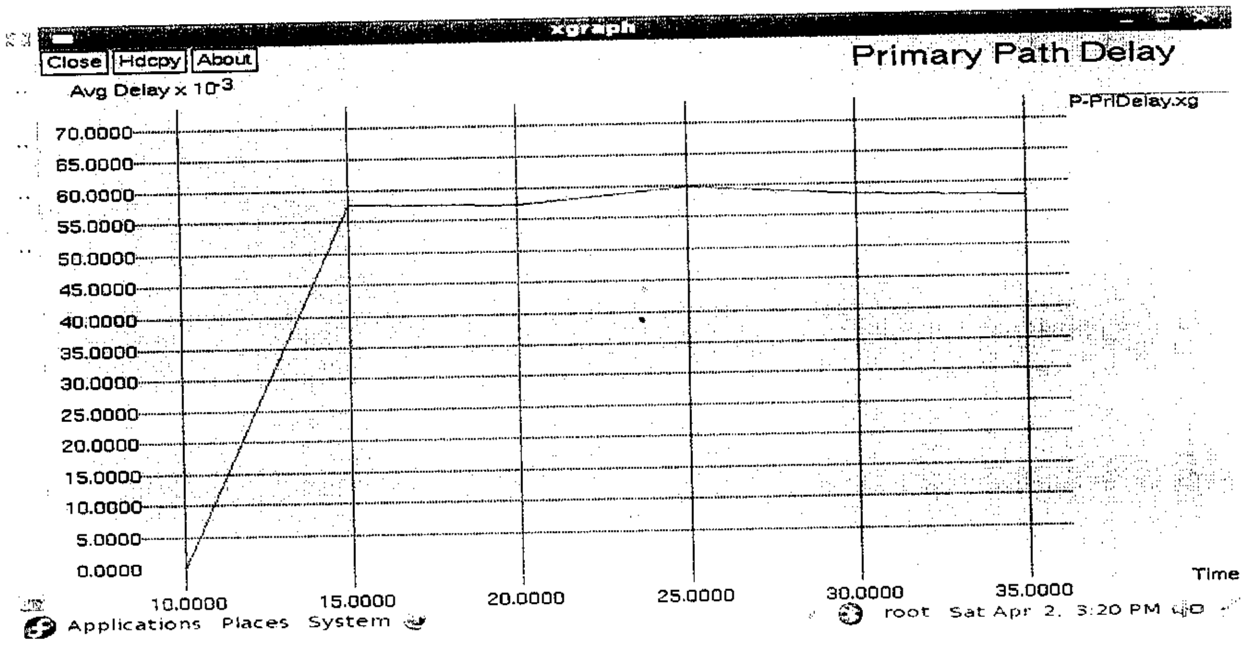
- **Primary Path Throughput**



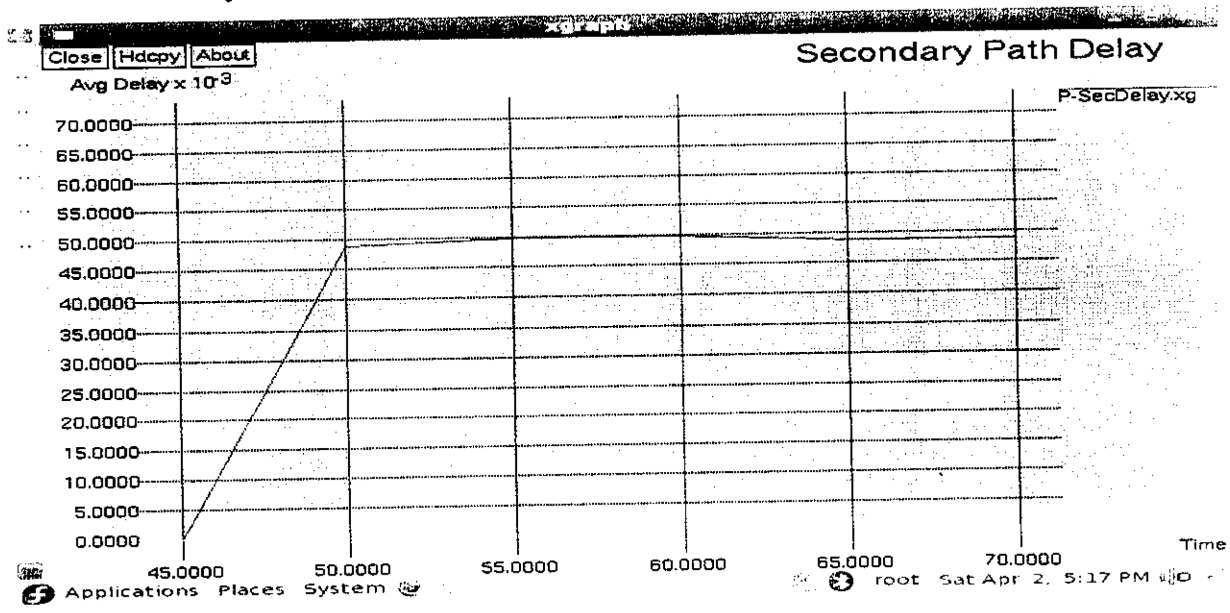
- **Secondary Path Throughput**



- **Primary Path Delay**



- **Secondary Path Delay**



REFERENCES

[1]. Akyildiz I.F., Sankarasubramaniam W. Su, Y. and Cayirci E, "A Survey on Sensor Networks", IEEE Communications Magazine, 2002.

[2]. Ewa Niewiadomska-Szynkiewicz, Piotr Kwa niewski, and Izabela Windyg, "Comparative Study of Wireless Sensor Networks Energy-Efficient Topologies and Power Save Protocols",In Journal of Telecommunication and Information Technology,March 2009.

[3]R Vidhyapriya , Dr P T Vanathi, "Energy Efficient Adaptive Multipath Routing for Wireless Sensor Networks",in IAENG International Journal of Computer Science,2007 .

[4]. Ankit tiwari, prasanna ballal, and frank l. Lewis, "Energy-Efficient Wireless Sensor Network Design and Implementation for Condition-Based Maintenance", ACM Trans. Sens. Netw. 3, 1, Article 1 (March 2007), 23 pages.

[5] Dimitrios J. Vergados & Nikolaos A. Pantazis , "Energy-Efficient Route Selection Strategies for Wireless Sensor Networks", in Mobile Netw Appl (2008) 13:285–296.

[6].Torsten Braun," Energy-Efficient TCP Operation in Wireless Sensor Networks".

[7] Sanatan Mohanty," Energy Efficient Routing Algorithms for Wireless Sensor Networks and Performance Evaluation of Quality of Service for IEEE 802.15.4 Network", thesis report,jan-2010

[8] Hongjoong Sin, Jangsoo Lee, Sungju Lee, Seunghwan Yoo, Sanghyuck Lee, Jaesik Lee, Yongjun Lee,and Sungchun Kim, "Agent-based Framework for Energy Efficiency in Wireless Sensor Networks",World Academy of Science.

[9] K. Akkaya, M. Younis, "An energy aware QoS routing protocol for wireless sensor networks", in: The Proceedings of the 23rd International Conference on Distributed Computing Systems Workshops, Providence, RI, USA, May 19_22,2003, pp. 710_715.

[10] E. Ayanoglu, E. Chih-Lin, R.D. Gitlin, J.E. Mazo," Diversity coding for transparent self-healing and fault-tolerant communication networks", IEEE Transactions on Communications 41 (11) (1993) 1677_1686.

[11] E. Felemban, C.G. Lee, E. Ekici, MMSPEED: "Multipath multispeed protocol for QoS guarantee of reliability and timelines in wireless sensor networks", IEEE Transactions on Mobile Computing 5 (6) (2006) 738_754.